Generative Models and Unsupervised Learning

Lecture notes from KAIST AI618

The students of AI618, edited by Jong Chul Ye

2026-10-16

Table of contents

Preface **1**
What the course is for . . . 1
How the book is organised . . . 1
How it was made . . . 2
Reading this book . . . 2
Contributing . . . 3
License . . . 3

Mathematical Preliminaries **5**
Metric Space . . . 5
Vector Space . . . 8
Banach and Hilbert Space . . . 11
Some Matrix Algebra . . . 16
Elements of Convex Optimization . . . 24

I. Optimal Transport, Adversarial and Latent Variable Models **37**

1. Statistical Distance and Optimal Transport **39**
1.1. Overview . . . 39
1.2. Notation . . . 40
1.3. Why the distance between distributions is the question . . . 40
1.4. The measure-theoretic minimum . . . 41
1.5. Metrics and divergences are not the same thing . . . 44
1.6. Divergences generated by a convex function . . . 44
1.7. Disjoint supports and the vanishing gradient . . . 51
1.8. Integral probability metrics . . . 54
1.9. Wasserstein distances . . . 57
1.10. Monge's formulation . . . 60
1.11. Kantorovich's relaxation . . . 65
1.12. Kantorovich duality . . . 67
1.13. Brenier's theorem and the Gaussian case . . . 70
1.14. Back to generative models . . . 75
1.15. Advanced topics . . . 76
1.16. Appendix: proof of the Radon–Nikodym theorem . . . 84
1.17. Further reading . . . 85

2. GAN and its Extensions **87**
2.1. Overview . . . 87
2.2. Notation . . . 87
2.3. The GAN minimax game . . . 88
2.4. Why the JSD saturates on disjoint supports . . . 90
2.5. The non-saturating loss . . . 90
2.6. Mode collapse . . . 92
2.7. f-divergences and f-GAN . . . 92
2.8. LS-GAN and the Pearson chi-squared divergence . . . 95
2.9. DCGAN: the architecture that made it train . . . 96
2.10. InfoGAN . . . 99
2.11. W-GAN . . . 100
2.12. Enforcing the Lipschitz constraint . . . 102
2.13. McGAN: mean and covariance feature matching . . . 104
2.14. Geometric GAN . . . 106
2.15. What the critic's loss does and does not tell you . . . 107
2.16. Worked example: JSD versus W1 on disjoint supports . . . 109
2.17. Connections and further reading . . . 110
2.18. Advanced topics . . . 112

3. StyleGAN, Normalization, Attention, and Image Translation **119**
3.1. Overview . . . 119
3.2. Notation . . . 119
3.3. Normalization as channel-wise transformation . . . 120
3.4. Style transfer: statistics, AdaIN, and transport . . . 122
3.5. Attention: transformation along the other axis . . . 124
3.6. Generators with style and content control . . . 126
3.7. Paired and unpaired image translation . . . 130
3.8. Multiple domains and multiple inputs . . . 137
3.9. Worked examples and reproducibility . . . 142
3.10. Connections and further reading . . . 143
3.11. Advanced topics . . . 144

4. VAE, beta-VAE and Normalizing Flow **153**
4.1. Overview . . . 153
4.2. Notation . . . 153
4.3. The evidence lower bound . . . 154
4.4. The general encoder . . . 156
4.5. The variational autoencoder . . . 158
4.6. The beta-VAE . . . 160
4.7. Normalizing and invertible flows . . . 162
4.8. NICE: additive coupling . . . 164
4.9. Real NVP: affine coupling . . . 166
4.10. Glow . . . 168

4.11. What the coupling construction costs . . . 170
4.12. Advanced topics . . . 170

II. Diffusion Models **179**

5. Self-supervised Denoising **181**
5.1. Overview . . . 181
5.2. Notation . . . 181
5.3. Denoising as an inverse problem . . . 182
5.4. Supervised denoising and why it is not enough . . . 186
5.5. Noise2Noise: learning from pairs of noisy images . . . 188
5.6. Noise2Self, J-invariance and Noise2Void . . . 191
5.7. SURE: a single image with a divergence penalty . . . 195
5.8. A side-by-side comparison . . . 197
5.9. Tweedie's formula . . . 198
5.10. Noise2Score . . . 204
5.11. Estimating the score from noisy data . . . 205
5.12. Where the independence assumption fails . . . 211
5.13. Connections and further reading . . . 212
5.14. Summary . . . 213
5.15. Advanced topics . . . 214

6. Denoising Diffusion Probabilistic Models **221**
6.1. Overview . . . 221
6.2. Notation . . . 222
6.3. Background: the Markov property and Gaussian tools . . . 223
6.4. Denoising diffusion probabilistic models . . . 225
6.5. DDIM: a non-Markovian forward process and deterministic sampling . . . 248
6.6. Reproducible numerical examples . . . 254
6.7. Summary . . . 264
6.8. Connections and further reading . . . 264
6.9. Appendix: proofs . . . 265

7. Score-based Generative Models with SDEs **271**

8. Conditional Diffusion Models **273**

9. Latent Diffusion **275**

10. Schrodinger Bridge Problems **277**

11. Acceleration: High-order Solvers and Consistency Models **279**

12. Image Editing and Inverse Problems **281**

13. Video and 3D Diffusion 283

III. Flow Models 285

14. Flow Matching 287

15. Rectified Flow 289

16. Relation between Flow and Diffusion Models 291

17. Probability Paths and the Kolmogorov Equation 293

IV. Discrete Diffusion 295

18. Discrete Flow Matching 297

19. Diffusion Language Models 299

20. Diffusion Duality 301

21. LLaDA and MMaDA 303

V. Alignment with Reinforcement Learning 305

22. RL Review: RLHF, DPO, DDPO and GRPO 307

23. Diffusion DPO, DDPO, DanceGRPO and FlowGRPO 309

24. DiffusionNFT, Reward Score Matching and Reward Adjoint Matching 311

References 313

Preface

This book is the lecture notes of **AI618, Generative Models and Unsupervised Learning**, a graduate course taught by Jong Chul Ye at KAIST. Every chapter began as a scribe note written by a student who attended the lecture, and was then merged, corrected, and edited into the chapter you are reading.

What the course is for

Generative modelling has accumulated a great many methods in a short time, and most treatments of it are surveys: here is a GAN, here is a VAE, here is a diffusion model, each introduced on its own terms. The course this book comes from is deliberately not that. It puts the emphasis on a **coherent framework that gives a unified perspective** on the whole subject, so that the methods appear as points in one landscape rather than as a list.

That means the mathematics is not optional background. Probability theory and **optimal transport** are developed because they are what makes the unity visible — the same transport problem that defines the Wasserstein distance in Chapter 1 returns as the Schrodinger bridge in the diffusion chapters and as the straight-line probability path in the flow chapters. Where a method is usually presented as a new idea, this book tries to show which older idea it is a special case of.

The course also insists on the implementations, from classical GANs through to diffusion and flow models, discrete diffusion, and alignment with RLHF. Where a chapter states a result, it should also say what you would have to write down to run it.

How the book is organised

Five parts, following the course:

Optimal Transport, Adversarial and Latent Variable Models. Statistical distances between distributions — f-divergences, integral probability metrics — and then optimal transport: Monge and Kantorovich, duality, Wasserstein distances. Then GANs read through that lens, from the original minimax objective out to f-GAN, W-GAN, McGAN, geometric GAN and LS-GAN, followed by style: style transfer and the StyleGAN family, unpaired translation with CycleGAN and its optimal transport form, and

multi-domain translation. Then the likelihood side: the ELBO, VAE and β-VAE, normalizing flows. The part closes on self-supervised denoising, where Tweedie's formula turns denoising into score estimation and opens the door to everything that follows.

Diffusion Models. DDPM and its variational bound, score-based models through the lens of stochastic differential equations, the probability flow ODE. Conditioning and guidance across two chapters, latent diffusion, Schrodinger bridges, acceleration by high-order solvers and consistency models, and the applications that made the family matter: editing, inverse problems, video and 3D.

Flow Models. Flow matching and its conditional form, rectified flow and straightening, the exact relationship to diffusion, and the Kolmogorov equations underneath both.

Discrete Diffusion. What has to change when the state space is categorical: discrete flow matching, diffusion language models, the duality with the continuous construction, and the recent large models built on it.

Alignment with Reinforcement Learning. RLHF, DPO and GRPO, their transfer to diffusion and flow models, and the newer objectives that avoid trajectory likelihoods altogether.

The through-line is worth stating once: **a score, a velocity field, and a transport map are three ways of writing the same object.** A reader who finishes the book should be able to move between them without thinking about it.

How it was made

Each lecture was written up independently by several students. Those notes were graded against a published rubric — coverage, technical correctness, exposition, figures, and derivations — with a language model producing three independent readings per note and a human reading every one before any grade was recorded. The model's score decided which notes were candidates for a chapter; it never decided a grade.

Chapters are **merged, not chosen.** With only a few independent write-ups of a lecture, the best of them still has gaps, and the grading pass produced an explicit list of what each note was missing. The merge starts from the strongest note and fills its gaps from the others.

Reading this book

Chapters follow the lecture order and the parts above. Notation is consistent within a chapter and mostly consistent across chapters; where a chapter departs from earlier usage it says so at the start. A chapter still marked *not yet written* is one whose scribe notes have not been merged.

The two references the course leans on throughout are Jong Chul Ye's *Geometry of Deep Learning* (Springer, 2022) and Lipman et al., *Flow Matching Guide and Code* (arXiv:2412.06264).

Contributing

Corrections are welcome from anyone, in or out of the course. Open an issue or a pull request. If you are reporting a mathematical error, quote the statement and say why it is wrong — that is more useful than a patch.

License

Licensed under CC BY 4.0. You may share and adapt this material, including commercially, with attribution. The students who wrote these notes are credited in each chapter.

Mathematical Preliminaries

In this chapter, we briefly review the basic mathematical concepts that are required to understand the materials of this book.

Metric Space

A metric space $(\mathcal{X}, d)$ is a set $\mathcal{X}$ together with a metric d on the set. Here, a *metric* is a function that defines a concept of distance between any two members of the set, which is formally defined as follows.

Definition 0.1. A metric on a set $\mathcal{X}$ is a function called the distance $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_+$, where $\mathbb{R}_+$ is the set of non-negative real numbers. For all $x, y, z \in \mathcal{X}$, this function is required to satisfy the following conditions:

1. $d(x, y) \geq 0$ (non-negativity).
2. $d(x, y) = 0$ if and only if $x = y$.
3. $d(x, y) = d(y, x)$ (symmetry).
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

Before going further it is worth seeing that the axioms are satisfied by more than the familiar distance, and that they do exclude things.

Example 0.1. The following are metric spaces.

1. The Euclidean metric $d(x, y) = \|x - y\|_2$ on $\mathbb{R}^n$.
2. The discrete metric on any set, $d(x, y) = 1$ if $x \neq y$ and 0 otherwise. Every subset is open under it, so it is a useful source of counterexamples.
3. The supremum metric on $C[a, b]$, the continuous functions on $[a, b]$, $d(f, g) = \sup_{t \in [a,b]} |f(t) - g(t)|$. Convergence in this metric is uniform convergence.
4. The metric induced by any norm on a vector space, $d(x, y) = \|x - y\|$, of which the l_p metrics of the l_p norms are the family we use most.

By contrast, the squared Euclidean distance $d(x, y) = \|x - y\|_2^2$ is *not* a metric: with $x = 0$, $y = 1$, $z = 2$ on the real line it gives $d(x, z) = 4$ while $d(x, y) + d(y, z) = 2$, so the triangle inequality fails. It is nonetheless a perfectly good objective function, which is why the distinction matters — being a useful measure of discrepancy and being a metric are different things.

A metric on a space induces topological properties like open and closed sets, which lead to the study of more abstract topological spaces. Specifically, about any point x in a metric space $\mathcal{X}$, we define the open ball of radius $r > 0$ about x as the set

$$B_r(x) = \{y \in \mathcal{X} : d(x, y) < r\}.$$

Using this, we have the formal definition of openness and closedness of a set.

Definition 0.2. A subset $U \subset \mathcal{X}$ is called open if for every $x \in U$ there exists an $r > 0$ such that $B_r(x)$ is contained in U. The complement of an open set is called closed.

A sequence (x_n) in a metric space $\mathcal{X}$ is said to converge to the limit $x \in \mathcal{X}$ if and only if for every $\varepsilon > 0$, there exists a natural number N such that $d(x_n, x) < \varepsilon$ for all $n > N$. A subset $\mathcal{S}$ of the metric space $\mathcal{X}$ is closed if and only if every sequence in $\mathcal{S}$ that converges to a limit in $\mathcal{X}$ has its limit in $\mathcal{S}$. In addition, a sequence of elements (x_n) is a Cauchy sequence if and only if (iff) for every $\varepsilon > 0$, there is some $N \geq 1$ such that

$$d(x_n, x_m) < \varepsilon, \quad \forall \quad m, n \geq N.$$

We are now ready to define the important concepts in metric spaces.

Definition 0.3. A metric space $\mathcal{X}$ is said to be complete if every Cauchy sequence converges to a limit; or if $d(x_n, x_m) \to 0$ as both n and m independently go to infinity, then there is some $y \in \mathcal{X}$ with $d(x_n, y) \to 0$.

Completeness is a property of the space, not of the metric alone. The rationals $\mathbb{Q}$ with $d(x, y) = |x - y|$ are not complete: the sequence $1, 1.4, 1.41, 1.414, ...$ is Cauchy in $\mathbb{Q}$ but its limit $\sqrt{2}$ is not a rational number. The same sequence in $\mathbb{R}$ converges. This is the sense in which $\mathbb{R}$ completes $\mathbb{Q}$, and it is why the iterative algorithms mentioned above are stated over $\mathbb{R}^n$ rather than over the numbers a computer can actually represent.

We turn now to maps between metric spaces. Continuity is defined first in the general form, and the Lipschitz condition afterwards as a stronger requirement.

Definition 0.4. Let $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ be metric spaces. A function $f : \mathcal{X} \mapsto \mathcal{Y}$ is *continuous at* $x_0 \in \mathcal{X}$ if for every $\varepsilon > 0$ there exists $\delta > 0$ such that

$$d_{\mathcal{X}}(x, x_0) < \delta \quad \Longrightarrow \quad d_{\mathcal{Y}}(f(x), f(x_0)) < \varepsilon. \tag{0.1}$$

It is *continuous* if it is continuous at every point, and *uniformly continuous* if the same δ works at every point, i.e. if for every $\varepsilon > 0$ there is a $\delta > 0$ such that $d_{\mathcal{Y}}(f(x), f(y)) < \varepsilon$ whenever $d_{\mathcal{X}}(x, y) < \delta$.

Two equivalent formulations are worth recording, because each is the convenient one in a different argument. First, f is continuous if and only if the preimage $f^{-1}(V)$ of every open $V \subset \mathcal{Y}$ is open in $\mathcal{X}$ — the formulation that survives when the metric is dropped and only the topology is kept. Second, f is continuous at x_0 if and only if $f(x_n) \to f(x_0)$ for every sequence $x_n \to x_0$, which is the one used to pass a limit through a function.

Definition 0.5. Given two metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$, where $d_{\mathcal{X}}$ denotes the metric on the set $\mathcal{X}$ and $d_{\mathcal{Y}}$ is the metric on set $\mathcal{Y}$, a function $f : \mathcal{X} \mapsto \mathcal{Y}$ is called Lipschitz continuous if there exists a real constant $K \geq 0$ such that, for all $x_1, x_2 \in \mathcal{X}$,

$$d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2).$$

Here, the constant K is often called the Lipschitz constant, and a function f with the Lipschitz constant K is called K-Lipschitz function.

The three conditions stand in a strict hierarchy,

$$\text{Lipschitz} \implies \text{uniformly continuous} \implies \text{continuous}, \tag{0.2}$$

and neither implication reverses. Taking $\delta = \varepsilon / K$ in Equation 0.1 proves the first, and the second is immediate from the definitions. For the failures: $f(x) = \sqrt{x}$ on $[0, 1]$ is uniformly continuous but not Lipschitz, since its slope is unbounded near 0; and $f(x) = x^2$ on $\mathbb{R}$ is continuous but not uniformly continuous, since the δ that works near the origin fails far from it. The distinction is not academic here — a Lipschitz gradient is exactly the hypothesis under which gradient descent with a fixed step size is guaranteed to descend.

Vector Space

A vector space $\mathcal{V}$ is a set that is closed under finite vector addition and scalar multiplication. In machine learning applications, the scalars are usually members of real or complex values, in which case $\mathcal{V}$ is called a vector space over real numbers, or complex numbers.

Stated in full, a vector space over a field $\mathbb{F}$ (for us $\mathbb{R}$ or $\mathbb{C}$) is a set $\mathcal{V}$ with an addition and a scalar multiplication satisfying, for all $u, v, w \in \mathcal{V}$ and $\alpha, \beta \in \mathbb{F}$:

1. $u + v = v + u$ and $(u + v) + w = u + (v + w)$;
2. there is a zero vector $\mathbf{0}$ with $u + \mathbf{0} = u$, and each u has an inverse $-u$ with $u + (-u) = \mathbf{0}$;
3. $\alpha(\beta u) = (\alpha\beta)u$ and $1u = u$;
4. $\alpha(u + v) = \alpha u + \alpha v$ and $(\alpha + \beta)u = \alpha u + \beta u$.

A *subspace* of $\mathcal{V}$ is a subset that is itself a vector space under the same operations. Verifying the eight axioms again is unnecessary: a non-empty $\mathcal{W} \subset \mathcal{V}$ is a subspace precisely when it is closed under linear combination, that is when $\alpha u + \beta v \in \mathcal{W}$ for all $u, v \in \mathcal{W}$ and all scalars α, β. Every subspace contains $\mathbf{0}$, which is the quickest way to rule one out.

For example, the Euclidean n-space $\mathbb{R}^n$ is called a real vector space, and $\mathbb{C}^n$ is called a complex vector space. In the n-dimensional Euclidean space $\mathbb{R}^n$, every element is represented by a list of n real numbers, addition is component-wise, and scalar multiplication is multiplication on each term separately. More specifically, we define a column n-real-valued vector x to be an array of n real numbers, denoted by

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^n,$$

where the superscript $^\top$ denotes the adjoint. Note that for a real vector, the adjoint is just a transpose. Then, the sum of the two vectors x and y, denoted by $x + y$, is defined by

$$x + y = \begin{bmatrix} x_1 + y_1 & x_2 + y_2 & \cdots & x_n + y_n \end{bmatrix}^\top.$$

Similarly, the scalar multiplication with a scalar $\alpha \in \mathbb{R}$ is defined by

$$\alpha x = \begin{bmatrix} \alpha x_1 & \alpha x_2 & \cdots & \alpha x_n \end{bmatrix}^\top.$$

In addition, we formally define the inner product and the norm in a vector space as follows.

Definition 0.6. Let $\mathcal{V}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{V}} : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$ is an inner product on $\mathcal{V}$ if:

1. Linear: $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_{\mathcal{V}} = \alpha_1 \langle f_1, g \rangle_{\mathcal{V}} + \alpha_2 \langle f_2, g \rangle_{\mathcal{V}}$ for all $\alpha_1, \alpha_2 \in \mathbb{R}$ and $f_1, f_2, g \in \mathcal{V}$.

2. Symmetric: $\langle f, g \rangle_{\mathcal{V}} = \langle g, f \rangle_{\mathcal{V}}$.

3. $\langle f, f \rangle_{\mathcal{V}} \geq 0$ and $\langle f, f \rangle_{\mathcal{V}} = 0$ if and only if $f = \mathbf{0}$.

If the underlying vector space $\mathcal{V}$ is obvious, we usually represent the inner product without the subscript $\mathcal{V}$, i.e. $\langle f, g \rangle$. For example, the inner product of the two vectors $f, g \in \mathbb{R}^n$ is defined as

$$\langle f, g \rangle = \sum_{i=1}^{n} f_i g_i = f^\top g.$$

Two nonzero vectors x, y are called *orthogonal* when

$$\langle x, y \rangle = 0,$$

which we denote as $x \perp y$. A vector x is orthogonal to a subset $\mathcal{S} \subset \mathcal{V}$, denoted by $x \perp \mathcal{S}$, if it is orthogonal to every element of $\mathcal{S}$. The orthogonal complement of $\mathcal{S}$, denoted by $\mathcal{S}^\perp$, consists of all vectors in $\mathcal{V}$ that are orthogonal to every vector in $\mathcal{S}$, i.e.

$$\mathcal{S}^\perp = \{x \in \mathcal{V} : \langle v, x \rangle = 0, \ \forall v \in \mathcal{S}\}.$$

Definition 0.7. A norm $\| \cdot \|$ is a real-valued function defined on the vector space that has the following properties:

1. $\|x\| \geq 0$, and $\|x\| = 0$ if and only if $x = \mathbf{0}$.

2. $\|\alpha x\| = |\alpha| \|x\|$ for any scalar α.

3. Triangular inequality: $\|x + y\| \leq \|x\| + \|y\|$ for any vectors x and y.

The lp Norms

The family we use throughout is the l_p norms, written $\|\cdot\|_p$. For $x \in \mathbb{R}^n$ and $p \geq 1$,

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \tag{0.3}$$

with the limiting case $\|x\|_\infty = \lim_{p\to\infty} \|x\|_p = \max_i |x_i|$. The cases $p = 1$, $p = 2$ and $p = \infty$ account for nearly every norm in this book: $\|\cdot\|_2$ is the induced norm of the Euclidean inner product, $\|\cdot\|_1$ is the sparsity-promoting surrogate of the discussion of convexity, and $\|\cdot\|_\infty$ is the dual norm of $\|\cdot\|_1$, as Table 3 records.

The first two norm axioms are immediate from Equation 0.3. The third is Minkowski's inequality, $\|x + y\|_p \leq \|x\|_p + \|y\|_p$ for $p \geq 1$, which is where the restriction on p enters.

It is worth seeing that the restriction is real rather than a technicality. Define $\|x\|_p$ by Equation 0.3 for some $0 < p < 1$ — say $p = 1/2$ — and take

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Then $\|x\|_{1/2} = \|y\|_{1/2} = 1$, while

$$\|x + y\|_{1/2} = \left(1^{1/2} + 1^{1/2}\right)^2 = 4 > 2 = \|x\|_{1/2} + \|y\|_{1/2},$$

so the triangle inequality fails and $\|\cdot\|_{1/2}$ is not a norm. The geometric statement of the same fact is that the set $\{x : \|x\|_p \leq 1\}$ is convex for $p \geq 1$ and non-convex for $p < 1$, its faces bending in towards the origin. Such $\|\cdot\|_p$ are called *quasi-norms*, and they satisfy the triangle inequality only up to a constant.

The non-convexity is exactly why they are used. The count of non-zero entries, written $\|x\|_0$, is the quantity one actually wants to minimise in a sparse recovery problem, and minimising it is combinatorial. The quasi-norms with $p < 1$ approximate $\|\cdot\|_0$ better than $\|\cdot\|_1$ does — at the price of the convexity that makes $\|\cdot\|_1$ tractable. That trade is the subject of a large literature, and the discussion of convexity explains what is lost when convexity goes.

From the inner product, we can obtain the so-called induced norm:

$$\|x\| = \sqrt{\langle x, x \rangle}.$$

Similarly, the definition of the metric in the metric space section informs us that a norm in a vector space $\mathcal{V}$ induces a metric, i.e.

$$d(x, y) = \|x - y\|, \quad x, y \in \mathcal{V}.$$

The norm and inner product in a vector space have special relations. For example, for any two vectors $x, y \in \mathcal{V}$, the following Cauchy–Schwarz inequality always holds:

$$|\langle x, y\rangle| \leq \|x\|\|y\|. \tag{0.4}$$

The proof is short enough to give, and the argument — minimise a quadratic that is non-negative by construction — recurs throughout the book.

Proof. If $y = \mathbf{0}$ both sides vanish, so assume $y \neq \mathbf{0}$ and take the real case. For every $t \in \mathbb{R}$,

$$0 \leq \|x - ty\|^2 = \|x\|^2 - 2t\langle x, y\rangle + t^2\|y\|^2.$$

The right-hand side is a quadratic in t that is never negative, so its discriminant cannot be positive: $4\langle x, y\rangle^2 - 4\|x\|^2\|y\|^2 \leq 0$, which is Equation 0.4. Equivalently, substituting the minimiser $t^\star = \langle x, y\rangle / \|y\|^2$ gives $0 \leq \|x\|^2 - \langle x, y\rangle^2 / \|y\|^2$. For a complex inner product the same argument runs with t chosen to align the phase of $\langle x, y\rangle$. This concludes the proof. □

Equality holds if and only if $\|x - t^\star y\| = 0$, that is if and only if x and y are linearly dependent. One consequence is that the induced norm really is a norm: expanding $\|x + y\|^2 = \|x\|^2 + 2\langle x, y\rangle + \|y\|^2$ and bounding the middle term by Equation 0.4 gives $\|x + y\|^2 \leq (\|x\| + \|y\|)^2$, which is the triangle inequality.

Banach and Hilbert Space

An *inner product space* is defined as a vector space that is equipped with an inner product. A *normed space* is a vector space on which a norm is defined. An inner product space is always a normed space since we can define a norm as $\|f\| = \sqrt{\langle f, f\rangle}$, which is often called the induced norm. Among the various forms of the normed space, one of the most useful normed spaces is the *Banach space*.

Definition 0.8. The Banach space is a complete normed space.

Here, the "completeness" is especially important from the optimization perspective, since most optimization algorithms are implemented in an iterative manner so that the final solution of the iterative method should belong to the underlying space $\mathcal{H}$. Recall that the convergence property is a property of a metric space. Therefore, the Banach space can be regarded as a vector space equipped with desirable properties of a metric space. Similarly, we can define the *Hilbert space*.

Definition 0.9. The Hilbert space is a complete inner product space.

We can easily see that the Hilbert space is also a Banach space thanks to the induced norm. The inclusion relationship between vector spaces, normed spaces, inner product spaces, Banach spaces and Hilbert spaces is illustrated in Figure 1.

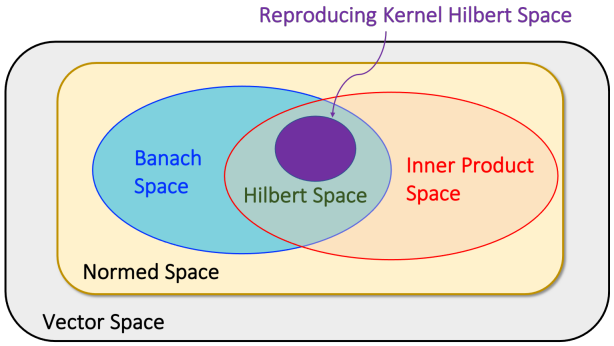


Figure 1.: RKHS, Hilbert space, Banach space, and vector space.

As shown in Figure 1, the Hilbert space has many nice mathematical structures such as inner product, norm, completeness, etc., so it is widely used in the machine learning literature. The following are well-known examples of Hilbert spaces:

- $l^2(\mathbb{Z})$: a function space composed of square summable discrete-time signals, i.e.

$$l^2(\mathbb{Z}) = \left\{ x = \{x_l\}_{l=-\infty}^{\infty} \;\middle|\; \sum_{l=-\infty}^{\infty} |x_l|^2 < \infty \right\}.$$

  Here, the inner product is defined as

$$\langle x, y \rangle_{\mathcal{H}} = \sum_{l=-\infty}^{\infty} x_l \overline{y_l}, \quad \forall x, y \in \mathcal{H}. \tag{0.5}$$

- $L^2(\mathbb{R})$: a function space composed of square integrable continuous-time signals, i.e.

$$L^2(\mathbb{R}) = \left\{ x(t) \;\middle|\; \int_{-\infty}^{\infty} |x(t)|^2 dt < \infty \right\}.$$

Here, the inner product is defined as

$$\langle x, y\rangle_{\mathcal{H}} = \int x(t)\overline{y(t)}dt. \tag{0.6}$$

Among the various forms of the Hilbert space, the reproducing kernel Hilbert space (RKHS) is of particular interest in the classical machine learning literature. Here, the readers are reminded that the RKHS is only a subset of the Hilbert space as shown in Figure 1, i.e. the Hilbert space is more general than the RKHS.

What distinguishes an RKHS is that pointwise evaluation is well behaved. In a general function space, two functions can be close in norm yet differ wildly at a given point — in $L^2(\mathbb{R})$ the value $f(x)$ at a single x is not even defined, since changing f on a set of measure zero changes nothing.

Definition 0.10. A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathbb{R}$ is a *reproducing kernel Hilbert space* if for every $x \in \mathcal{X}$ the evaluation functional $\delta_x : f \mapsto f(x)$ is bounded, that is if there is a constant C_x with $|f(x)| \leq C_x \|f\|_{\mathcal{H}}$ for all $f \in \mathcal{H}$.

Boundedness is the whole of the definition, and the familiar machinery follows from it. By the Riesz representation theorem a bounded linear functional is an inner product against some element, so for each x there exists $k_x \in \mathcal{H}$ with

$$f(x) = \langle f, k_x\rangle_{\mathcal{H}}, \qquad \forall f \in \mathcal{H}, \tag{0.7}$$

which is the *reproducing property*. Writing $k(x, y) := k_y(x) = \langle k_y, k_x\rangle_{\mathcal{H}}$ defines the *reproducing kernel*, a symmetric function that is positive definite: for any points $x_1, \dots, x_m$ and any coefficients $c_1, \dots, c_m$,

$$\sum_{i,j} c_i c_j k(x_i, x_j) = \left\| \sum_i c_i k_{x_i} \right\|^2_{\mathcal{H}} \geq 0.$$

The Moore–Aronszajn theorem is the converse, and it is what makes the notion useful: every symmetric positive-definite k arises this way from exactly one RKHS. One may therefore choose a kernel and never construct the space at all, which is the content of the "kernel trick" — the Gaussian kernel $k(x, y) = \exp(-\|x - y\|^2/2\sigma^2)$ being the standard example, with the polynomial kernel $k(x, y) = (\langle x, y\rangle + c)^d$ the other common one. The Gaussian kernel's RKHS is infinite-dimensional, so a finite inner product computation stands in for one in a space that cannot be written down.

Basis and Frames

The set of vectors $\{x_1, \cdots, x_k\}$ is said to be *linearly independent* if a *linear combination* denoted by

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k = \mathbf{0}$$

implies that

$$\alpha_i = 0, \quad i = 1, \cdots, k.$$

The set of all vectors *reachable* by taking linear combinations of vectors in a set $\mathcal{S}$ is called the *span* of $\mathcal{S}$. For example, if $\mathcal{S} = \{x_i\}_{i=1}^k$, then we have

$$\text{span}(\mathcal{S}) = \left\{ \sum_{i=1}^{k} \alpha_i x_i, \forall \alpha_i \in \mathbb{R} \right\}.$$

A set $\mathcal{B} = \{b_i\}_{i=1}^m$ of elements (vectors) in a vector space $\mathcal{V}$ is called a basis, if every element of $\mathcal{V}$ may be written in a unique way as a linear combination of elements of $\mathcal{B}$, that is, for all $f \in \mathcal{V}$, there exists unique coefficients $\{c_i\}$ such that

$$f = \sum_{i=1}^{m} c_i b_i.$$

A set $\mathcal{B}$ is a basis of $\mathcal{V}$ if and only if every element of $\mathcal{B}$ is linearly independent and $\text{span}(\mathcal{B}) = \mathcal{V}$. The coefficients of this linear combination are referred to as expansion coefficients, or coordinates on $\mathcal{B}$ of the vector. The elements of a basis are called basis vectors. Any two bases of the same vector space have the same number of elements, and that number is the *dimension* $\dim \mathcal{V}$. So the phrase "m-dimensional space" means exactly that every basis of it has m elements, and a set of more than m vectors in it is necessarily dependent. For example, when $\mathcal{V} = \mathbb{R}^2$, the following two sets are some examples of a basis:

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}, \quad \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}.$$

For function spaces, the number of basis vectors can be infinite. For example, for the space V_T composed of periodic functions with the period of T, the following complex sinusoidals constitute its basis:

$$B = \{\varphi_n(t)\}_{n=-\infty}^{\infty}, \quad \varphi_n(t) = e^{i\frac{2\pi nt}{T}},$$

so that any function $x(t) \in V_T$ can be represented by

$$x(t) = \sum_{n=-\infty}^{\infty} a_n \varphi_n(t),$$

where the expansion coefficient is given by

$$a_n = \frac{1}{T} \int_T x(t) \varphi_n^*(t) dt.$$

In fact, this basis expansion is often called the *Fourier series.*

Unlike the basis, which leads to the unique expansion, the frame is composed of redundant basis vectors, which allows multiple representations. For example, consider the following frame in $\mathbb{R}^2$:

$$\{v_1, v_2, v_3\} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Then, we can easily see that the frame allows multiple representations of, for example, $x = [2, 3]^\top$ as shown in the following:

$$x = 2v_1 + 3v_2 = v_2 + 2v_3.$$

Frames can also be extended to deal with function spaces, in which case the number of frame elements is infinite.

Formally, a set of functions

$$\Phi = [\phi_k]_{k \in \Gamma} = \begin{bmatrix} \cdots & \phi_{k-1} & \phi_k & \cdots \end{bmatrix}$$

in a Hilbert space H is called a *frame* if it satisfies the following inequality:

$$\alpha \|f\|^2 \leq \sum_{k \in \Gamma} |\langle f, \phi_k \rangle|^2 \leq \beta \|f\|^2, \quad \forall f \in H, \tag{0.8}$$

where $\alpha, \beta > 0$ are called the frame bounds. If $\alpha = \beta$, then the frame is said to be tight. In fact, an orthonormal basis is a special case of a tight frame with $\alpha = \beta = 1$, which is Parseval's identity. A general basis need not be a frame at all.

The frame inequality says a redundant expansion loses nothing; what makes it usable is that the expansion can be inverted. Collect the frame vectors as the *analysis operator* $\Phi : \mathcal{H} \mapsto l^2(\Gamma)$, $(\Phi f)_k = \langle f, \phi_k \rangle$, whose adjoint Φ^* is the *synthesis operator* $\Phi^* c = \sum_k c_k \phi_k$. Their composition is the *frame operator*

$$S = \Phi^* \Phi, \qquad Sf = \sum_{k \in \Gamma} \langle f, \phi_k \rangle \phi_k. \tag{0.9}$$

In this notation the frame inequality reads $\alpha I \preceq S \preceq \beta I$, so the lower bound $\alpha > 0$ is exactly the statement that S is invertible. Setting $\tilde{\phi}_k := S^{-1}\phi_k$, the *canonical dual frame*, and applying S^{-1} to Equation 0.9 gives the reconstruction formula

$$f = \sum_{k \in \Gamma} \langle f, \phi_k \rangle \tilde{\phi}_k = \sum_{k \in \Gamma} \langle f, \tilde{\phi}_k \rangle \phi_k. \tag{0.10}$$

For a tight frame $S = \alpha I$, so $\tilde{\phi}_k = \alpha^{-1}\phi_k$ and the formula collapses to $f = \alpha^{-1} \sum_k \langle f, \phi_k \rangle \phi_k$ — no inverse to compute. This is why tight frames are the ones worth constructing, and with $\alpha = 1$ it is the ordinary orthonormal expansion.

Note that Equation 0.10 is not the only expansion of f: when the frame is redundant the coefficients are not unique, and the canonical dual is the choice that gives the coefficient sequence of least l^2 norm. Choosing a different expansion — the sparsest rather than the smallest — is the starting point of sparse coding.

Some Matrix Algebra

In the following, we introduce some matrix algebra that is useful in understanding the materials in this book.

A *matrix* is a rectangular array of numbers, denoted by an upper case letter, say A. A matrix with m rows and n columns is called an $m \times n$ matrix given by

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

The k-th column of matrix A is often denoted by a_k. The maximal number of linearly independent columns of A is called the *rank* of the matrix A. It is easy to show that

$$\text{Rank}(A) = \dim \text{ span } ([a_1, \cdots, a_n]) .$$

The trace of a square matrix $A \in \mathbb{R}^{n \times n}$, denoted $\text{Tr}(A)$ is defined to be the sum of elements on the main diagonal (from the upper left to the lower right) of A:

$$\text{Tr}(A) = \sum_{i=1}^{n} a_{ii} \ .$$

Definition 0.11. The range space of a matrix $A \in \mathbb{R}^{m\times n}$, denoted by $\mathcal{R}(A)$, is defined by $\mathcal{R}(A) := \{Ax \mid \forall x \in \mathbb{R}^n\}$.

Definition 0.12. The null space of a matrix $A \in \mathbb{R}^{m\times n}$, denoted by $\mathcal{N}(A)$, is defined by $\mathcal{N}(A) := \{x \in \mathbb{R}^n \mid Ax = \mathbf{0}\}$.

A subset of a vector space is called a *subspace* if it is closed under both addition and scalar multiplication. We can easily see that the range and null spaces are subspaces. Moreover, we can show the following fundamental property:

$$\mathcal{R}(A)^\perp = \mathcal{N}(A^\top), \quad \mathcal{N}(A)^\perp = \mathcal{R}(A^\top). \tag{0.11}$$

Both identities follow from the definitions in one line. A vector y is orthogonal to $\mathcal{R}(A)$ exactly when $\langle y, Ax\rangle = 0$ for all x, and since $\langle y, Ax\rangle = \langle A^\top y, x\rangle$, that holds for all x precisely when $A^\top y = \mathbf{0}$. The second identity is the first applied to $A^\top$.

These are the *four fundamental subspaces* of $A \in \mathbb{R}^{m\times n}$. Writing $r = \operatorname{rank}(A)$, their dimensions are

$$\dim \mathcal{R}(A) = \dim \mathcal{R}(A^\top) = r, \qquad \dim \mathcal{N}(A) = n - r, \qquad \dim \mathcal{N}(A^\top) = m - r. \tag{0.12}$$

The first equality — that the column rank and the row rank agree — is what lets one speak of *the* rank at all. The second is the rank–nullity theorem, $\dim \mathcal{R}(A) + \dim \mathcal{N}(A) = n$: every dimension of the domain is either mapped somewhere or collapsed.

Together with Equation 0.11, Equation 0.12 says that A splits its domain and codomain into orthogonal pieces,

$$\mathbb{R}^n = \mathcal{R}(A^\top) \oplus \mathcal{N}(A), \qquad \mathbb{R}^m = \mathcal{R}(A) \oplus \mathcal{N}(A^\top), \tag{0.13}$$

and that A maps the r-dimensional $\mathcal{R}(A^\top)$ bijectively onto the r-dimensional $\mathcal{R}(A)$ while sending $\mathcal{N}(A)$ to zero. This is the picture behind the pseudo-inverse of the pseudo-inverse: inverting the bijective part is the most that can be asked of a matrix that is not invertible.

If a vector space $\mathcal{V}$ is Hilbert space, then it is known that for a subspace $\mathcal{S} \in \mathcal{V}$ and the vector $y \in \mathcal{V}$, the point in $\mathcal{S}$ that is closest to y exists and is unique, and given by

$$\hat{y} = \mathcal{P}_{\mathcal{S}} y$$

where $\mathcal{P}_{\mathcal{S}}$ is the *projector* associated with the subspace $\mathcal{S}$. In particular, if the subspace $\mathcal{S}$ has a basis B, then the projector for $\mathcal{S}$ is given by

$$\mathcal{P}_{\mathcal{S}} = B(B^\top B)^{-1}B^\top.$$

The eigen-decomposition of a square matrix is defined as follows.

Definition 0.13. A (nonzero) vector $v \in \mathbb{C}^n$ is an eigenvector of a square matrix $A \in \mathbb{C}^{n\times n}$ if it satisfies the linear equation

$$Av = \lambda v,$$

where λ is a scalar, termed the eigenvalue corresponding to v.

Equivalently, λ is an eigenvalue when $A - \lambda I$ is singular, so the eigenvalues are the roots of the characteristic polynomial $\det(A - \lambda I) = 0$. An $n \times n$ matrix therefore has n eigenvalues over $\mathbb{C}$, counted with multiplicity.

If A has n linearly independent eigenvectors, collecting them as the columns of V and the eigenvalues into a diagonal Λ turns the n equations $Av_i = \lambda_i v_i$ into the single statement $AV = V\Lambda$, and V is invertible, so

$$A = V\Lambda V^{-1}. \tag{0.14}$$

Such an A is called *diagonalisable*. Not every matrix is: the independent eigenvectors may be fewer than n, as for $\left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$, which has the single eigenvalue 1 and only one eigenvector direction.

The symmetric case is both better behaved and the one we need. The *spectral theorem* states that a real symmetric $A = A^\top$ — or a Hermitian $A = A^H$ in the complex case — is always diagonalisable, its eigenvalues are real, and its eigenvectors can be chosen orthonormal, so that Equation 0.14 becomes

$$A = U\Lambda U^\top, \qquad U^\top U = I. \tag{0.15}$$

This is the statement that gives meaning to the ordering of eigenvalues used for positive semidefinite operators later in this chapter, and it connects the eigen-decomposition to the SVD defined next: $A^\top A$ is symmetric and positive semidefinite for any A, its eigenvectors are the right singular vectors of A, and its eigenvalues are the squared singular values, $\lambda_i(A^\top A) = \sigma_i(A)^2$. The matrix norms defined below are written in exactly that form.

We now define the singular value decomposition (SVD) of A.

Theorem 0.1. *If $A \in \mathbb{C}^{m\times n}$ is a rank r matrix, then there exist matrices $U \in \mathbb{C}^{m\times r}$ and $V \in \mathbb{C}^{n\times r}$ such that $U^H U = V^H V = I_r$ and $A = U\Sigma V^H$, where ^{H} denotes the conjugate transpose, I_r is the $r \times r$ identity matrix and Σ is an $r \times r$ diagonal matrix whose diagonal entries, called* singular values*, satisfy*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0.$$

The decomposition can be written as

$$A = \begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_r \end{bmatrix} \begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix}^\top = \sum_{k=1}^{r} \sigma_k u_k v_k^\top,$$

where u_k and v_k are called left singular vectors and right singular vectors, respectively.

Using the SVD, we can easily show the following:

$$\mathcal{P}_{\mathcal{R}(A)} = UU^\top, \quad \mathcal{P}_{\mathcal{R}(A^\top)} = VV^\top.$$

The Moore–Penrose Pseudo-inverse

The decomposition Equation 0.13 says that A is a bijection from $\mathcal{R}(A^\top)$ onto $\mathcal{R}(A)$, and inverting that bijection is what the pseudo-inverse does. In terms of the compact SVD $A = U\Sigma V^\top$,

$$A^\dagger := V\Sigma^{-1}U^\top = \sum_{k=1}^{r} \frac{1}{\sigma_k} v_k u_k^\top, \tag{0.16}$$

which is well defined because Σ collects only the r non-zero singular values. When A is invertible, $A^\dagger = A^{-1}$.

Applying Equation 0.16 to the projectors obtained above gives

$$A^\dagger A = VV^\top = \mathcal{P}_{\mathcal{R}(A^\top)}, \qquad AA^\dagger = UU^\top = \mathcal{P}_{\mathcal{R}(A)}, \tag{0.17}$$

so neither product is the identity unless the corresponding null space is trivial. These two identities are half of the four *Penrose conditions*, which characterise $A^\dagger$ uniquely as the matrix X satisfying

$$AXA = A, \qquad XAX = X, \qquad (AX)^\top = AX, \qquad (XA)^\top = XA. \tag{0.18}$$

The reason the pseudo-inverse appears throughout inverse problems is the following. Consider $\min_x \|Ax - b\|_2^2$, which need not have a unique solution and, if $b \notin \mathcal{R}(A)$, has no exact one. The residual is smallest when Ax is the projection of b onto $\mathcal{R}(A)$, and among all x achieving it, adding any element of $\mathcal{N}(A)$ changes nothing. Then

$$x^\star = A^\dagger b$$

is the unique minimiser lying in $\mathcal{R}(A^\top)$, i.e. the least-squares solution of smallest norm. This is the first example in the book of a problem with no unique answer being made well posed by choosing among the solutions, and Equation 0.16 shows the price: the small singular values sit in a denominator, so a σ_k near zero amplifies whatever noise b carries in the direction u_k. Every regularisation scheme in later chapters is, in one way or another, a response to that.

Using the SVD, we can define the matrix norm. Among the various forms of matrix norms for a matrix $X \in \mathbb{R}^{n\times n}$, the spectral norm $\|X\|_2$ and the nuclear norm $\|X\|_*$ are quite often used, which are defined by

$$\begin{aligned}\|X\|_2 &= \sigma_{\max}(X) = (\lambda_{\max}(X^\top X))^{1/2},\\ \|X\|_* &= \sum_i \sigma_i(X) = \sum_i (\lambda_i(X^\top X))^{1/2},\end{aligned}$$

where $\sigma_{\max}(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the largest singular value and eigenvalue, respectively.

The following matrix inversion lemma is quite useful.

Lemma 0.1.

$$\begin{aligned}(I + UCV)^{-1} &= I - U\left(C^{-1} + VU\right)^{-1} V,\\ (A + UCV)^{-1} &= A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1}.\end{aligned} \tag{0.19}$$

Kronecker Product

In mathematics, the Kronecker product, sometimes denoted by $\otimes$, is an operation on two matrices of arbitrary size resulting in a block matrix. The formal definition is given as follows.

Definition 0.14. If A is an $m \times n$ matrix and B is a $p \times q$ matrix, then the Kronecker product $A \otimes B$ is the $pm \times qn$ block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

The Kronecker product has many important properties, which can be exploited to simplify many matrix-related operations. Some of the basic properties are provided in the following lemma. The proofs of the lemmas are straightforward, which can easily be found from a standard linear algebra textbook.

Lemma 0.2.

$$\begin{aligned} A \otimes (B + C) &= A \otimes B + A \otimes C. \\ (B + C) \otimes A &= B \otimes A + C \otimes A. \\ A \otimes B &\neq B \otimes A. \\ (A \otimes B) \otimes C &= A \otimes (B \otimes C). \\ (A \otimes B)^\top &= A^\top \otimes B^\top. \\ (A \otimes B)^{-1} &= A^{-1} \otimes B^{-1}. \end{aligned} \tag{0.20}$$

Lemma 0.3. *If A, B, C and D are matrices of such a size that one can form the matrix products AC and BD, then*

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \tag{0.21}$$

One of the important usages of the Kronecker product comes from the vectorization operation of a matrix. For this we first define the following two operations.

Definition 0.15. If $A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \in \mathbb{R}^{m \times n}$, then

$$\mathrm{Vec}(A) = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \in \mathbb{R}^{mn},$$

$$\mathrm{UnVec}(\mathrm{Vec}(A)) = \mathrm{UnVec}\left(\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \right) = A.$$

From these definitions, we can obtain the following two lemmas which will be extensively used here.

Lemma 0.4. *For the matrices A, B, C with appropriate sizes, we have*

$$\mathrm{Vec}(CAB) = (B^\top \otimes C)\mathrm{Vec}(A), \tag{0.22}$$

where $\mathrm{Vec}(\cdot)$ *is the column-wise vectorization operation.*

Lemma 0.5. *For the vectors $x \in \mathbb{R}^m, y \in \mathbb{R}^n$, we have*

$$\mathrm{Vec}(xy^\top) = (y \otimes I_m)x,$$

where I_m denotes the $m \times m$ identity matrix.

Proof. By plugging $C = I_m, A = x$ and $B = y^\top$ into Equation 0.22, we conclude the proof. □

Matrix and Vector Calculus

In computing a derivative of a scalar, vector, or matrix with respect to a scalar, vector, or matrix, we should be consistent with the notation. In fact, there are two different conventions: numerator layout and denominator layout. For example, for a given scalar y and a column vector $x = [x_1, \cdots, x_n]^\top \in \mathbb{R}^n$, the numerator layout has the following convention:

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \cdots & \frac{\partial y}{\partial x_n} \end{bmatrix}, \quad \frac{\partial x}{\partial y} = \begin{bmatrix} \frac{\partial x_1}{\partial y} \\ \vdots \\ \frac{\partial x_n}{\partial y} \end{bmatrix},$$

implying that the number of the row follows that of the numerator. On the other hand, the denominator layout notation provides

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}, \quad \frac{\partial x}{\partial y} = \begin{bmatrix} \frac{\partial x_1}{\partial y} & \cdots & \frac{\partial x_n}{\partial y} \end{bmatrix},$$

where the number of resulting rows follows that of the denominator. Either layout convention is okay, but we should be consistent in using the convention.

Here, we will follow the denominator layout convention. The main motivation for using the denominator layout is from the derivative with respect to the matrix. More specifically, for a given scalar c and a matrix $W \in \mathbb{R}^{m \times n}$, according to the denominator layout, we have

$$\frac{\partial c}{\partial W} = \begin{bmatrix} \frac{\partial c}{\partial w_{11}} & \cdots & \frac{\partial c}{\partial w_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial c}{\partial w_{m1}} & \cdots & \frac{\partial c}{\partial w_{mn}} \end{bmatrix} \in \mathbb{R}^{m \times n}. \tag{0.23}$$

Furthermore, this notation leads to the following familiar result:

$$\frac{\partial a^\top x}{\partial x} = \frac{\partial x^\top a}{\partial x} = a. \tag{0.24}$$

Accordingly, for a given scalar c and a matrix $W \in \mathbb{R}^{m\times n}$, we can show that

$$\frac{\partial c}{\partial W} := \text{UnVec}\left(\frac{\partial c}{\partial \text{Vec}(W)}\right) \in \mathbb{R}^{m\times n},$$

in order to be consistent with Equation 0.23. Under the denominator layout notation, for given vectors $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$, the derivative of a vector with respect to a vector is given by

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix} \in \mathbb{R}^{m\times n}. \tag{0.25}$$

Then, the chain rule can be specified as follows:

$$\frac{\partial c(g(u))}{\partial x} = \frac{\partial u}{\partial x}\frac{\partial g(u)}{\partial u}\frac{\partial c(g)}{\partial g}. \tag{0.26}$$

Equation 0.24 also leads to

$$\frac{\partial Ax}{\partial x} = A^\top. \tag{0.27}$$

Finally, the following result is useful.

Lemma 0.6. *Let $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. Then, we have*

$$\frac{\partial Ax}{\partial \text{Vec}(A)} = x \otimes I_m.$$

Proof. Using Lemma 0.4, we have $Ax = \text{Vec}(Ax) = (x^\top \otimes I_m)\text{Vec}(A)$. Thus,

$$\begin{aligned} \frac{\partial Ax}{\partial \text{Vec}(A)} &= \frac{\partial (x^\top \otimes I_m)\text{Vec}(A)}{\partial \text{Vec}(A)} \\ &= (x^\top \otimes I_m)^\top \\ &= x \otimes I_m, \end{aligned}$$

where we use Equation 0.24 and Equation 0.20 for the second and the third equalities, respectively. Q.E.D. □

Lemma 0.7. *Let x, a and B denote vectors and a matrix with appropriate sizes, respectively. Then, we have*

$$\frac{\partial x^\top a}{\partial x} = \frac{\partial a^\top x}{\partial x} = a,$$
$$\frac{\partial x^\top B x}{\partial x} = (B + B^\top)x.$$

For a given scalar function $\ell : x \in \mathbb{R}^n \mapsto \mathbb{R}$, the derivative is often called the gradient, which can be represented by the denominator layout:

$$\nabla \ell := \frac{\partial \ell}{\partial x} \in \mathbb{R}^n.$$

Elements of Convex Optimization

Some Definitions

Let $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ be non-empty sets. The identity operator on $\mathcal{H}$ is denoted by $\mathcal{I}$, i.e. $\mathcal{I}x = x, \forall x \in \mathcal{H}$. Let $\mathcal{D} \subset \mathcal{H}$ be a non-empty set. The set of the *fixed points* of an operator $\mathcal{T} : D \mapsto D$ is denoted by

$$\mathrm{Fix}\mathcal{T} = \{x \in \mathcal{D} \mid \mathcal{T}x = x\}.$$

Let $\mathcal{X}$ and $\mathcal{Y}$ be real normed vector space. As a special case of an operator, we define a set of *linear operators*:

$$\mathcal{B}(\mathcal{X}, \mathcal{Y}) = \{\mathcal{T} : \mathcal{X} \mapsto \mathcal{Y} \mid \mathcal{T} \text{ is linear and continuous}\}$$

and we write $\mathcal{B}(\mathcal{X}) = \mathcal{B}(\mathcal{X}, \mathcal{X})$. Let $f : \mathcal{X} \mapsto [-\infty, \infty]$ be a function. The *domain* of f is

$$\mathrm{dom}f = \{x \in \mathcal{X} | f(x) < \infty\},$$

the *graph* of f is

$$\mathrm{gra}f = \{(x, y) \in \mathcal{X} \times \mathbb{R} | f(x) = y\},$$

and the *epigraph* of f is

$$\text{epi}f = \{(x, y) : x \in \mathcal{X}, y \in \mathbb{R}, y \geq f(x)\}.$$

The *indicator function* $\iota_{\mathcal{C}} : \mathcal{X} \mapsto [-\infty, \infty]$ of $\mathcal{C} \subset \mathcal{X}$ is defined as

$$\iota_{\mathcal{C}}(x) = \begin{cases} 0, & \text{if } x \in \mathcal{C}, \\ \infty, & \text{otherwise.} \end{cases}$$

We often use another definition of the indicator function:

$$\chi_{\mathcal{C}}(x) = \begin{cases} 1, & \text{if } x \in \mathcal{C}, \\ 0, & \text{otherwise.} \end{cases}$$

The *support function* of a set $\mathcal{C}$ is defined as

$$S_{\mathcal{C}}(x) = \sup\{\langle x, y\rangle | y \in \mathcal{C}\}.$$

An *affine function* is denoted by

$$x \mapsto \mathcal{T}x + b, \quad x \in \mathcal{X}, y \in \mathcal{Y}, \mathcal{T} \in \mathcal{B}(\mathcal{X}, \mathcal{Y}).$$

A function f is called *lower semicontinuous* at x_0 if for every $\varepsilon > 0$ there exists a neighbourhood $\mathcal{U}$ of x_0 such that $f(x) \geq f(x_0) - \varepsilon$ for all $x \in \mathcal{U}$. This is expressed as

$$\liminf_{x \to x_0} f(x) \geq f(x_0).$$

A function is lower semicontinuous if and only if all of its lower level sets $\{x \in \mathcal{X} : f(x) \leq \alpha\}$ are closed. Alternatively, f is lower semicontinuous if and only if the epigraph of f is closed. A function is *proper* if $-\infty \notin f(\mathcal{X})$ and $\text{dom}f \neq \emptyset$.

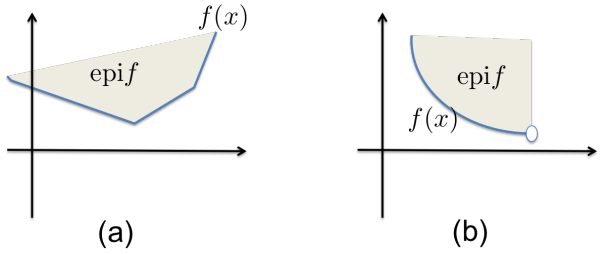


Figure 2.: Epigraphs for (a) a lower semicontinuous function, and (b) a function which is not lower semicontinuous.

A self-adjoint operator $\mathcal{A} : \mathcal{H} \mapsto \mathcal{H}$ is positive semidefinite if and only if

$$\langle x, \mathcal{A}x \rangle \geq 0, \quad \forall x \in \mathcal{H}.$$

A self-adjoint operator $\mathcal{A} : \mathcal{H} \mapsto \mathcal{H}$ is positive definite if and only if

$$\langle x, \mathcal{A}x \rangle > 0, \quad \forall x \in \mathcal{H} \setminus \{\mathbf{0}\}.$$

For simplicity, we denote $\mathcal{A} \succeq 0$ (resp. $\mathcal{A} \succ 0$) for positive semidefinite (resp. positive definite) operators. If $\mathcal{A} : \mathbb{C}^n \mapsto \mathbb{C}^n$, then $\mathbb{S}^n_{++}$ and $\mathbb{S}^n_{+}$ denote the set of $n \times n$ positive definite and positive semidefinite matrices, respectively. Here, the eigenvalues of positive semidefinite (resp. positive definite) are all real and non-negative (resp. positive).

Convex Sets, Convex Functions

A function $f(x)$ is a *convex function* if $\mathrm{dom} f$ is a convex set and

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$$

for all $x_1, x_2 \in \mathrm{dom} f$, $0 \leq \theta \leq 1$. A *convex set* is a set that contains every line segment between any two points in the set (see Figure 3). Specifically, a set $\mathcal{C}$ is convex if $x_1, x_2 \in \mathcal{C}$, then $\theta x_1 + (1-\theta)x_2 \in C$ for all $0 \leq \theta \leq 1$. The relation between a convex function and a convex set can also be stated using its epigraph. Specifically, a function $f(x)$ is convex if and only if its epigraph $\mathrm{epi} f$ is a convex set.

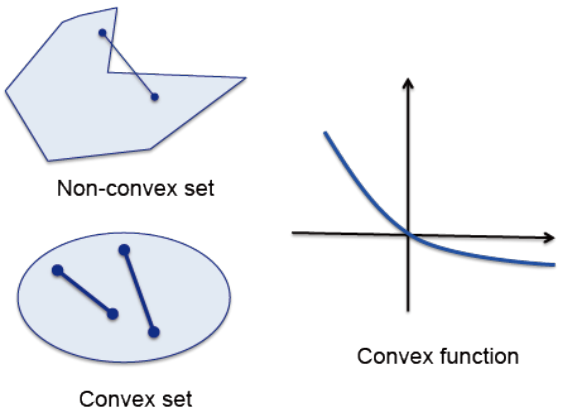


Figure 3.: A convex set and a convex function.

Convexity is preserved under various operations. For example, if $\{f_i\}_{i \in I}$ is a family of convex functions, then, $\sup_{i \in I} f_i$ is convex. In addition, a set of convex functions is closed under addition and multiplication by strictly positive real numbers. Moreover, the limit

point of a convergent sequence of convex functions is also convex. Important examples of convex functions are summarized in Table 1.

Table 1.: Examples of convex functions.

Name	$f(x)$
exponential	$e^{ax}, \quad \forall a \in \mathbb{R}$
quadratic over linear	$x^2/y, \quad (x,y) \in \mathbb{R} \times \mathbb{R}_{++}$
Huber function	$\begin{cases} \lvert x\rvert^2/2\mu, & \text{if } \lvert x\rvert < \mu \\ \lvert x\rvert - \mu/2, & \text{if } \lvert x\rvert \geq \mu \end{cases}$
relative entropy	$y\log y - y\log x, \quad (x,y) \in \mathbb{R}_{++} \times \mathbb{R}_{++}$
indicator function	$\iota_{\mathcal{C}}(x), \quad \mathcal{C}$: convex set
support function	$S_{\mathcal{C}}(x) = \sup\{\langle x, y\rangle \lvert y \in C\}$
distance to a set	$d(x, \mathcal{S}) = \inf_{y\in\mathcal{S}} \lVert x - y\rVert$
affine function	$Tx + b, \quad x \in \mathbb{R}^n.$
quadratic function	$x^\top Qx/2, \quad x \in \mathbb{R}^n, Q \in \mathbb{S}_+$
p-norms	$\lVert x\rVert_p = \left(\sum_i \lvert x_i\rvert^p\right)^{1/p}, p \geq 1$
l_∞-norm	$\lVert x\rVert_\infty = \max_i \lvert x_i\rvert$
max function	$\max\{x_1, \cdots, x_n\}$
log-sum-exponential	$\log\left(\sum_{i=1}^n e^{x_i}\right), x = (x_1, \cdots, x^{(k)}) \in \mathbb{R}^n.$
Gaussian data fidelity	$\lVert y - Ax\rVert^2, \quad x \in \mathcal{H}$
Poisson data fidelity	$\langle 1, Ax\rangle - \langle y, \log(Ax)\rangle, \quad x \in \mathbb{R}^n, 1 = (1, \cdots, 1) \in \mathbb{R}^n$
spectral norm	$\lVert X\rVert_2 = \sigma_{\max}(X) = (\lambda_{\max}(X^\top X))^{1/2}, X \in \mathbb{R}^{n\times n}$
nuclear norm	$\lVert X\rVert_* = \sum_i \sigma_i(X) = \sum_i (\lambda_i(X^\top X))^{1/2}, X \in \mathbb{R}^{n\times n}$

A function f is *concave* if $-f$ is convex. It is easy to show that an affine function $f(x) = Ax + b$ is both convex and concave. Examples of concave functions that are often used in this textbook can be found in Table 2.

Table 2.: Examples of concave functions.

Name	$f(x)$
powers	$x^p, \quad 0 \leq p \leq 1, \quad x \in \mathbb{R}_{++}$
geometric mean	$\left(\prod_{i=1}^n x_i\right)^{\frac{1}{n}}$
logarithm	$\log x, \quad x \in \mathbb{R}_{++}$
log determinant	$\log\det(X), \quad X \in \mathbb{S}_{++}$

Subdifferentials

The *directional derivative* of f at $x \in \mathrm{dom} f$ in the direction of $y \in \mathcal{H}$ is defined by

$$f'(x;y) = \lim_{\alpha \downarrow 0} \frac{f(x+\alpha y) - f(x)}{\alpha} \tag{0.28}$$

if the limit exists. If the limit exists for all $y \in \mathcal{H}$, then one says that f is *Gâteaux differentiable* at x. Suppose $f'(x;\cdot)$ is linear and continuous on $\mathcal{H}$. Then, there exist a unique gradient vector $\nabla f(x) \in \mathcal{H}$ such that

$$f'(x;y) = \langle y, \nabla f(x) \rangle, \quad \forall y \in \mathcal{H}.$$

If a function is differentiable, the convexity of a function can easily be checked using the first- and second-order differentiability, as stated in the following:

Theorem 0.2. *Let $f : \mathcal{H} \mapsto (-\infty, \infty]$ be proper. Suppose that* $\mathrm{dom} f$ *is open and convex, and f is Gateaux differentiable on* $\mathrm{dom} f$*. Then, the following are equivalent:*

1. *f is convex.*
2. *(First-order): $f(y) \geq f(x) + \langle y - x, \nabla f(x) \rangle, \quad \forall x, y \in \mathcal{H}$.*
3. *(Monotonicity of gradient): $\langle y - x, \nabla f(y) - \nabla f(x) \rangle \geq 0, \quad \forall x, y \in \mathcal{H}$.*

Theorem 0.2 uses the gradient only. When f is twice differentiable there is a second characterisation, and it is usually the one that settles a concrete case.

Theorem 0.3. *Let f be twice differentiable with* $\mathrm{dom} f$ *open and convex. Then f is convex if and only if its Hessian is positive semidefinite everywhere,*

$$\nabla^2 f(x) \succeq 0, \qquad \forall x \in \mathrm{dom} f. \tag{0.29}$$

On the real line this reads $f''(x) \geq 0$, the familiar test. Two cautions about the strict version. If $\nabla^2 f(x) \succ 0$ everywhere then f is strictly convex, but the converse fails: $f(x) = x^4$ is strictly convex while $f''(0) = 0$. And strict convexity is weaker than *strong convexity*, which asks for $\nabla^2 f(x) \succeq \mu I$ with $\mu > 0$ — a uniform curvature bound that strict convexity does not provide, and the hypothesis under which gradient descent converges at a linear rate.

As an example, take the quadratic $f(x) = \frac{1}{2} x^\top Q x - b^\top x$ with Q symmetric. Here $\nabla f(x) = Qx - b$ and $\nabla^2 f(x) = Q$, so f is convex exactly when $Q \succeq 0$, strongly convex exactly when $Q \succ 0$, and in that case the minimiser is the solution of $Qx = b$. By Equation 0.15 the eigenvalues of Q decide all three questions at once.

If the convergence in Equation 0.28 is uniform with respect to y on bounded sets, i.e.

$$\lim_{0\neq y\to 0} \frac{f(x+y) - f(x) - \langle y, \nabla f(x)\rangle}{\|y\|} = 0, \tag{0.30}$$

then f is *Frechet differentiable* and $\nabla f(x)$ is called the *Frechet gradient* of f at x. If f is differentiable and convex, then it is clear that

$$x \in \arg\min f \Leftrightarrow \nabla f(x) = 0.$$

However, if f is not differentiable, we need a more general framework to characterize the minimizers. The *sub-differential* of f is a set-valued operator defined as

$$\partial f(x) = \{u \in \mathcal{H} : f(y) \geq f(x) + \langle y - x, u\rangle, \quad \forall y \in \mathcal{H}\}. \tag{0.31}$$

The elements of sub-differential $\partial f(x)$ are called *sub-gradients* of f at x. Another important role of the subdifferentials comes from *Fermat's rule* that characterizes the global minimizers (Figure 4):

Theorem 0.4. *Let $f : \mathcal{H} \mapsto (-\infty, \infty]$ be proper. Then,*

$$\arg\min f = \text{zer}\partial f = \{x \in \mathcal{H} \mid 0 \in \partial f(x)\}. \tag{0.32}$$

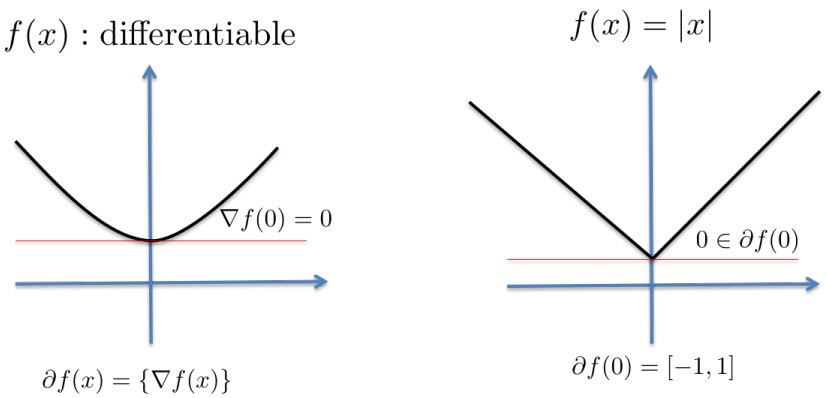


Figure 4.: Fermat's rule for the global minimizer.

Convex Conjugate

A *convex conjugate* or *convex dual* is very important concept for both classical and modern convex optimization techniques. Formally, the *conjugate* function $f^* : \mathcal{H} \mapsto [-\infty, \infty]$ of a function $f : \mathcal{H} \mapsto [-\infty, \infty]$ is defined as

$$f^*(u) = \sup_{x \in \mathcal{H}} \{\langle u, x \rangle - f(x)\}. \tag{0.33}$$

The transform in Equation 0.33 is often called *Legendre-Fenchel transform.*

Figure 5(a) shows a geometric interpretation of the convex conjugate when $\mathcal{H} = \mathbb{R}$. For example, when $f(x) = x^2 - x$, the convex conjugate $f^*(u)$ at $u = 1$ is the maximum difference between $g(x) = x$ and $f(x) = x^2 - x$, which occurs at $x = 1$ in this example. The difference is also equal to the magnitude of the y-intercept of the supporting hyperplane of $f(x)$ at $x = 1$. Figure 5(b) shows another intuitive example. Here, $f(x) = bx + c$. In this case, the difference between the line $g_1(x) = u_1 x$ and $f(x)$ becomes infinite at $x \to -\infty$. Similarly, the difference between the line $g_2(x) = u_2 x$ and $f(x)$ becomes infinite at $x \to \infty$. Only when $u = b$ does the maximum distance become finite, and it is then equal to $-c$. Therefore, the convex conjugate of $f(x) = bx + c$ is

$$f^*(u) = \begin{cases} -c, & u = b, \\ \infty, & u \neq b. \end{cases}$$

Table 3 summarizes these findings for a variety of functions that are often used in biomedical imaging applications.

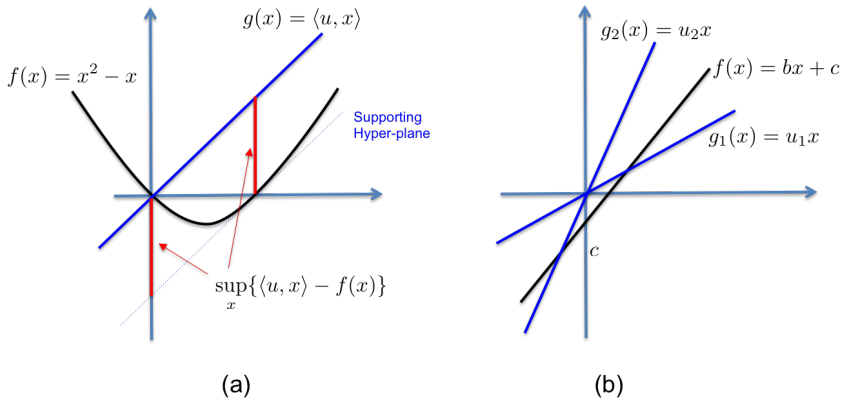


Figure 5.: (a) Geometry of convex conjugate. (b) Examples of finding convex conjugate for $f(x) = bx + c$.

Table 3.: Examples of convex conjugate pairs used often in biomedical imaging problems.

$f(x)$	$\mathrm{dom} f$	$f^*(u)$	$\mathrm{dom} f^*$
$f(ax)$	D	$f^*(u/a)$	D
$f(x+b)$	D	$f^*(u) - \langle b, u\rangle$	D
$af(x), a>0$	D	$af^*(u/a)$	D
$bx + c$	D	$\begin{cases} -c, & u = b \\ +\infty, & u \neq b \end{cases}$	$\{b\}$
$1/x$	$\mathbb{R}_{++}$	$-2\sqrt{-u}$	$-\mathbb{R}_{+}$
$-\log x$	$\mathbb{R}_{++}$	$-(1+\log(-u))$	$-\mathbb{R}_{++}$
$x\log x$	$\mathbb{R}_{+}$	e^{u-1}	$\mathbb{R}$
$\sqrt{1+x^2}$	$\mathbb{R}$	$-\sqrt{1-u^2}$	$[-1,1]$
e^x	$\mathbb{R}$	$u\log(u) - u$	$\mathbb{R}_{+}$
$\log(1+e^x)$	$\mathbb{R}$	$u\log(u) + (1-u)\log(1-u)$	$[0,1]$
$-\log(1-e^x)$	$\mathbb{R}_{--}$	$u\log(u) - (1+u)\log(1+u)$	$\mathbb{R}_{+}$
$\frac{\lvert x\rvert^p}{p},\ p>1$	$\mathbb{R}$	$\frac{\lvert u\rvert^q}{q},\ \frac{1}{p}+\frac{1}{q}=1$	$\mathbb{R}$
$\lVert x\rVert_1$	$\mathbb{R}^n$	$\begin{cases} 0, & \lVert u\rVert_\infty \leq 1 \\ \infty & \lVert u\rVert_\infty > 1 \end{cases}$	$\{u \in \mathbb{R}^n : \lVert u\rVert_\infty \leq 1\}$
$\langle a, x\rangle + b$	$\mathbb{R}^n$	$\begin{cases} -b, & u = a \\ \infty, & u \neq a \end{cases}$	$\{a\} \subset \mathbb{R}^n$
$\frac{1}{2}x^\top Q x, \quad Q \in \mathbb{S}_{++}$	$\mathbb{R}^n$	$\frac{1}{2}u^\top Q^{-1} u$	$\mathbb{R}^n$
$\iota_C(x)$	C	$S_C(u)$	$\mathcal{H}$
$\log\left(\sum_{i=1}^n e^{x_i}\right)$	$\mathbb{R}^n$	$\sum_{i=1}^n u_i \log u_i, \quad \sum_{i=1}^n u_i = 1$	$\mathbb{R}^n_{+}$
$-\log\det X^{-1}$	$\mathbb{S}^n_{++}$	$\log\det(-U)^{-1} - n$	$-\mathbb{S}^n_{++}$

Here, $D \subset \mathcal{H}$ and we use the interpretation $0\log 0 = 0$. It is clear that f^* is convex since f^* is a point-wise supremum of a convex function of y. In general, if $f : \mathcal{H} \mapsto [-\infty, \infty]$, then the following hold:

1. For $\alpha \in \mathbb{R}_{++}$, we have

$$(\alpha f)^* = \alpha f^*(\cdot/\alpha). \tag{0.34}$$

2. *Fenchel–Young inequality*:

$$f(x) + f^*(y) \geq \langle y, x\rangle, \quad \forall x, y \in \mathcal{H}. \tag{0.35}$$

3. Let f, g be proper functions from $\mathcal{H}$ to $(-\infty, \infty]$. Then,

$$f(x) + g(x) \geq -f^*(u) - g^*(-u), \quad \forall x, u \in \mathcal{H}. \tag{0.36}$$

If f is convex, proper, and lower semicontinuous, then the following properties hold:

$$\begin{aligned} f^{**} &= f, \\ y \in \partial f(x) \iff \quad f(x) + f^*(y) &= \langle x, y \rangle \iff x \in \partial f^*(y). \end{aligned} \tag{0.37}$$

Lagrangian Dual Formulation

Perhaps one of the most important uses of convex conjugate is to obtain the dual formulation. More specifically, for a given *primal problem* (P),

$$(P): \quad \min_{x \in \mathcal{H}} f(x) + g(x),$$

we can obtain the associated *dual problem* using Equation 0.36:

$$(D): \quad -\min_{u \in \mathcal{H}} f^*(u) + g^*(-u).$$

The gap between the primal and dual problem is called the *duality gap*.

Example 0.2 (Dual for Composite Function)**.** For the given primal problem:

$$(P): \quad \min_{x \in \mathbb{R}^n} f(x) + g(Ax), \tag{0.38}$$

with $A \in \mathbb{R}^{m \times n}$, the dual problem is given by

$$(D): -\min_{u \in \mathbb{R}^m} f^*(A^\top u) + g^*(-u).$$

Proof. Note that (P) is equivalent to the following constraint minimization problem:

$$\begin{aligned} &\min_{x,y} f(x) + g(y) \\ &\text{subject to} Ax = y, \end{aligned}$$

which provides

$$\begin{aligned} \min_{x \in \mathbb{R}^n} f(x) + g(Ax) &\geq \min_{x,y} f(x) + g(y) + u^\top (Ax) - u^\top y \\ &= \min_x \{f(x) + (A^\top u)^\top x\} + \min_y \{g(y) - u^\top y\} \\ &= -f^*(-A^\top u) - g^*(u), \end{aligned}$$

where the first line drops the constraint $Ax = y$, so the relaxation is a lower bound for every u, and the second line holds with equality because the minimisation separates in x and y.

Since this holds for every u, taking the supremum gives

$$\max_{u\in\mathbb{R}^m} -f^*(-A^\top u) - g^*(u) = -\min_{u\in\mathbb{R}^m} f^*(-A^\top u) + g^*(u),$$

and substituting $u \leftarrow -u$ yields the dual problem

$$-\min_{u\in\mathbb{R}^m} f^*(A^\top u) + g^*(-u).$$

This concludes the proof. □

Example 0.3 (Quadratic Programming under Affine Constraint)**.** Consider the following optimization problem:

$$P : \min \frac{1}{2}x^\top x \quad \text{subject to} \quad b = Ax$$

with $A \in \mathbb{R}^{m\times n}$. Now, we define $\mathcal{C} = \{0\}$ such that $b - Ax \in \mathcal{C}$. Then, the original minimization problem becomes

$$\begin{aligned} &\min_{x,y} \iota_{\mathcal{C}}(y) + \frac{1}{2}x^\top x \\ &\text{subject to} y = b - Ax. \end{aligned}$$

Therefore, we have

$$\begin{aligned} \min_x \iota_{\mathcal{C}}(b - Ax) + \frac{1}{2}x^\top x &\geq \min_{x,y} \iota_{\mathcal{C}}(y) + \frac{1}{2}x^\top x + u^\top(b - Ax - y) \\ &\geq \min_y \iota_{\mathcal{C}}(y) - u^\top y + \min_x \frac{1}{2}x^\top x - u^\top Ax + u^\top b \\ &\geq \min_{y\in\{0\}} -u^\top y + \min_x \frac{1}{2}x^\top x - u^\top Ax + u^\top b \\ &= -\frac{1}{2}u^\top AA^\top u + u^\top b, \end{aligned}$$

where the last equality comes from $x = A^\top u$ at the minimizer. Hence, the dual problem becomes

$$D : \min_{u\in\mathbb{R}^m} \frac{1}{2}u^\top AA^\top u - u^\top b.$$

Why is this dual formulation useful? Suppose that A is highly ill-posed, say that $n = 1000$ and $m = 1$. Then, the dual problem (D) is a one-dimensional problem which is computationally much less expensive than the primal problem (P) of the dimension $n = 1000$. After the dual solution $\widehat{u}$ is obtained, the primal solution is just $\widehat{x} = A^\top \widehat{u}$.

We formally define a Lagrangian dual problem.

Definition 0.16. Suppose that a primal problem is given by

$$\begin{aligned} &\min_x f_0(x) \\ \text{subject to} \quad &f_i(x) \leq 0, \quad i = 1, \cdots, n, \\ &h_i(x) = 0, \quad i = 1, \cdots, p. \end{aligned}$$

Then, the associated Lagrangian dual problem is defined by

$$\begin{aligned} &\max_{\alpha, \nu} g(\alpha, \nu) \\ \text{subject to} \quad &\alpha \geq \mathbf{0}, \end{aligned}$$

where $\alpha = [\alpha_1, \cdots, \alpha_n]$ and $\nu = [\nu_1, \cdots, \nu_p]$ are referred to as the *dual variables* or *Lagrangian multipliers*, $\alpha \geq \mathbf{0}$ implies that each element is non-negative, and the *Lagrangian dual function* $g(\alpha, \nu)$ is defined by

$$g(\alpha, \nu) := \inf_x \left\{ f_0(x) + \sum_{i=1}^{n} \alpha_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x) \right\}.$$

Two properties hold with no assumption on the primal problem at all. First, g is concave, being a pointwise infimum of functions that are affine in (α, ν) — so the dual problem is a concave maximisation, hence convex, even when the primal is not. Second, *weak duality*: for any feasible x and any $\alpha \geq \mathbf{0}$,

$$g(\alpha, \nu) \;\leq\; f_0(x) + \sum_{i=1}^{n} \alpha_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x) \;\leq\; f_0(x), \tag{0.39}$$

where the first inequality is the definition of the infimum and the second holds because $\alpha_i f_i(x) \leq 0$ and $h_j(x) = 0$ at a feasible x. Taking the best on each side gives $d^\star \leq p^\star$, and the difference $p^\star - d^\star \geq 0$ is the *duality gap*. This is what makes the dual useful even when it is not tight: any feasible dual point certifies a lower bound on the primal optimum.

One of the important findings in convex optimization theory is that if the primal problem is convex *and a constraint qualification holds* — for example Slater's condition, which

asks that some strictly feasible point exist, i.e. a point x with $f_i(x) < 0$ for all i and $h_j(x) = 0$ for all j — then we have the following strong duality:

$$g(\alpha^*, \nu^*) = f_0(x^*), \tag{0.40}$$

where x^* and α^*, ν^* are the optimal solutions for the primal and dual problems, respectively. i.e. the duality gap is zero. Often, the dual formulation is easier to solve than the primal problem.

When strong duality holds and the functions are differentiable, the primal and dual optima are characterised by the *Karush–Kuhn–Tucker* conditions: at x^* and (α^*, ν^*),

$$\begin{aligned} f_i(x^*) \le 0, \quad h_j(x^*) = 0 \quad & \text{(primal feasibility)}, \\ \alpha_i^* \ge 0 \quad & \text{(dual feasibility)}, \\ \alpha_i^* f_i(x^*) = 0 \quad & \text{(complementary slackness)}, \\ \nabla f_0(x^*) + \sum_i \alpha_i^* \nabla f_i(x^*) + \sum_j \nu_j^* \nabla h_j(x^*) = \mathbf{0} \quad & \text{(stationarity)}. \end{aligned} \tag{0.41}$$

Complementary slackness is the informative one: a constraint that is inactive at the optimum carries a zero multiplier, so the multipliers say which constraints are actually binding. For a convex problem satisfying Slater's condition the KKT conditions are necessary and sufficient, which turns a constrained problem into a system of equations and inequalities. Note that stationarity is the constrained analogue of $\nabla f = \mathbf{0}$, and that Fermat's rule of the subdifferential section generalises it to the non-differentiable case.

There is also an interesting geometric interpretation, which will be explained later in this book.

Written by Jong Chul Ye.

Part I.

Optimal Transport, Adversarial and Latent Variable Models

1. Statistical Distance and Optimal Transport

1.1. Overview

A generative model does not produce a single number that we can compare against a label. It produces a whole distribution, and training it means making that distribution resemble the distribution the data came from. So before any architecture is discussed, we need an answer to a prior question: given two probability distributions over the same space, what does it mean to say that one is close to the other, and how close?

This lecture gives two answers, and they are genuinely different. The first compares distributions pointwise, through the ratio of their densities; this produces the family of f-divergences, of which the Kullback–Leibler and Jensen–Shannon divergences are the members that generative modelling actually uses. The second compares them by asking how much work it takes to carry the mass of one distribution onto the other; this produces optimal transport and the Wasserstein distances. The distinction is not decorative. We will see that when two distributions have disjoint supports — which, for a generative model of images, is the normal situation rather than an edge case — every f-divergence is constant and therefore carries no gradient, while the Wasserstein distance still varies smoothly. That single fact is the reason the GAN literature moved from the original minimax objective to the Wasserstein one, and it is the reason this material comes first in the course.

By the end the reader should be able to state and prove what an f-divergence is and when it is finite, derive the Kullback–Leibler and Jensen–Shannon divergences from their generators, recognise integral probability metrics as the dual-norm view of the same problem and estimate one of them from samples, set up both Monge's and Kantorovich's transport problems and say why the first can fail in three different ways, prove Kantorovich duality in the discrete case and read off from it both the Kantorovich–Rubinstein form of W_1 and the c-cyclical monotonicity of optimal plans, and identify the optimal transport map between two Gaussians as an instance of Brenier's theorem.

This chapter assumes the metric space, convexity and duality material of the preliminaries chapter and does not restate it.

1.2. Notation

Symbol	Meaning
$\mathcal{X}, \mathcal{Y}$	Polish spaces; the data space and the latent space
$\mathcal{P}(\mathcal{X})$	Borel probability measures on $\mathcal{X}$
μ, ν	probability measures; μ is the source, ν the target
P, Q	the same measures when densities or masses are meant
$\nu \ll \mu$	ν is absolutely continuous with respect to μ
$d\nu/d\mu$	Radon–Nikodym derivative of ν with respect to μ
$T_{\#}\mu$	push-forward of μ under the map T
$\Pi(\mu, \nu)$	couplings of μ and ν
$c(x, y)$	transport cost from x to y
φ^c	c-transform of the potential φ
$\mathrm{Lip}_1(\mathcal{X})$	1-Lipschitz real functions on $\mathcal{X}$
G_θ	generator with parameters θ
p_{data}	the unknown data distribution

Throughout, log is the natural logarithm and the convention $0 \log 0 = 0$ is in force, justified by continuity of $t \log t$ at $t = 0$.

1.3. Why the distance between distributions is the question

Fix a latent space $\mathcal{Y}$ carrying a simple reference measure ν — in practice $\nu = \mathcal{N}(0, I)$ — and a data space $\mathcal{X}$. A generator is a measurable map $G_\theta : \mathcal{Y} \to \mathcal{X}$. It does not model a density directly. What it gives us is a procedure for drawing samples: take $z \sim \nu$ and return $G_\theta(z)$. The distribution of the result is the image of ν under G_θ, written

$$\mu_\theta = G_{\theta\#}\nu, \tag{1.1}$$

and made precise in Definition 1.2 below and drawn in Figure 1.1. Training the generator means choosing θ so that μ_θ resembles p_{data}, which we only ever see through a finite sample. So we need a function $D(\mu_\theta, p_{\mathrm{data}})$ that is small exactly when the two agree, and — this is the part that is easy to forget — whose gradient in θ points somewhere useful when they do not.

Written out, the whole of generative modelling in this chapter is

$$\min_{\theta} D(\mu_\theta, \mu) \qquad \text{subject to} \qquad \mu_\theta = G_{\theta\#}\nu, \tag{1.2}$$

the form in which the lecture states it (Ye 2026a, 51) and the one that organises the geometric reading of GANs due to Lei and co-authors (Lei et al. 2017). Everything that follows is a choice of D in Equation 1.2, and the chapter is in effect a catalogue of those choices and of what each one does to the gradient.

There is an obstruction, and it is structural rather than accidental. The support of μ_θ is contained in $G_\theta(\mathcal{Y})$, the image of a space of dimension $\dim \mathcal{Y}$, which in practice is a hundred or so. The data are believed to lie near a manifold of similarly low dimension inside a pixel space of dimension 10^5 or more. Two such sets, both of measure zero in the ambient space, generically fail to overlap. Any discrepancy built out of the ratio of two densities is therefore being asked to compare $0/0$ almost everywhere, and we will prove in Section 1.7 that it responds by saturating at a constant. Keeping that obstruction in view is what organises the rest of the chapter.

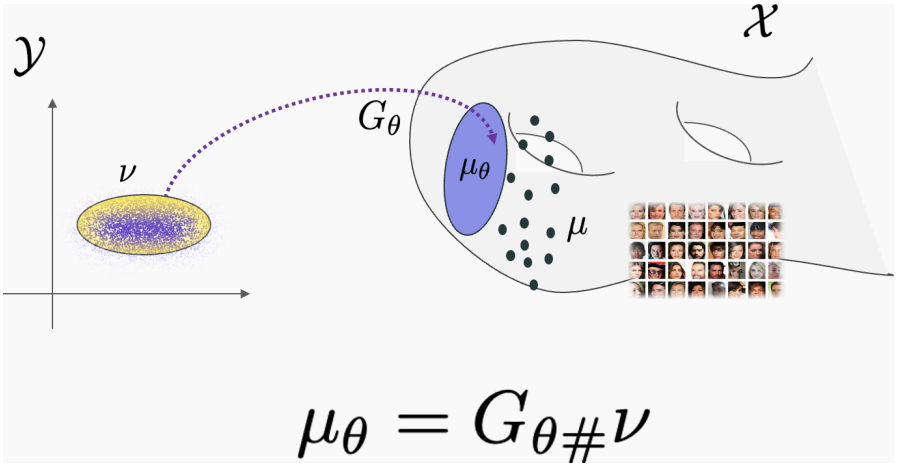


Figure 1.1.: The setting, and the obstruction. The latent measure ν on $\mathcal{Y}$ is carried by the generator to $\mu_\theta = G_{\theta\#}\nu$, which lives on the image $G_\theta(\mathcal{Y})$ — the shaded patch inside $\mathcal{X}$, not all of it. The data measure μ sits elsewhere on the same low-dimensional set. Two such supports generically fail to overlap, which is why a discrepancy built from a ratio of densities has nothing to compare. From the course slides (Ye 2026a, 10).

1.4. The measure-theoretic minimum

This section fixes three definitions that the rest of the chapter uses without further comment. It is deliberately brief; the preliminaries chapter treats measure and integration

properly.

Definition 1.1 (Absolutely continuous measure)**.** Let μ and ν be measures on a measurable space $(\Omega, \mathcal{F})$. We say ν is absolutely continuous with respect to μ, written $\nu \ll \mu$, if $\mu(A) = 0$ implies $\nu(A) = 0$ for every $A \in \mathcal{F}$.

The condition says that ν cannot put mass where μ puts none. It is the exact hypothesis under which a density of ν with respect to μ can exist, which is the content of the next theorem. Figure 1.2 contrasts the two cases.

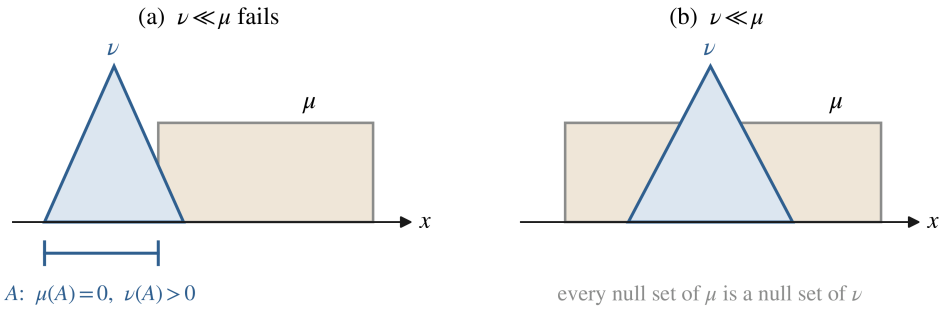


Figure 1.2.: Absolute continuity. In (a) the measure ν charges a region that μ does not, so no density of ν against μ can exist there and $\nu \not\ll \mu$. In (b) the support of ν sits inside that of μ and $\nu \ll \mu$.

Theorem 1.1 (Radon–Nikodym)**.** *Let λ and ν be σ-finite measures on $(\Omega, \mathcal{F})$ with $\lambda \ll \nu$. Then there is a non-negative $\mathcal{F}$-measurable function g on Ω, unique up to ν-null sets, such that*

$$\lambda(A) = \int_A g \, d\nu, \qquad A \in \mathcal{F}. \tag{1.3}$$

The function g is written $d\lambda/d\nu$. A proof is given in Section 1.16 rather than here, because the argument is self-contained but long enough to interrupt the development. An ordinary probability density function is the special case in which ν is Lebesgue measure: writing $p = dP/dx$ recovers $P(A) = \int_A p(x)\, dx$.

Definition 1.2 (Push-forward of a measure)**.** Let $(\mathcal{X}_1, \mathcal{B}_1)$ and $(\mathcal{X}_2, \mathcal{B}_2)$ be measurable spaces, μ a measure on the first, and $T : \mathcal{X}_1 \to \mathcal{X}_2$ measurable. The push-forward $T_{\#}\mu$ is the measure on $\mathcal{B}_2$ defined by

$$(T_{\#}\mu)(A) = \mu(T^{-1}(A)) = \mu(\{x : T(x) \in A\}). \tag{1.4}$$

Read Equation 1.4 as a statement about sampling: if $x \sim \mu$ then $T(x) \sim T_{\#}\mu$. For a generator this is exactly Equation 1.1, and it is worth noting what the definition does not require. There is no inverse of T anywhere in it — only the preimage of a set, which exists for any map — and no density. A generator that collapses a whole region of latent space to one image is perfectly well described.

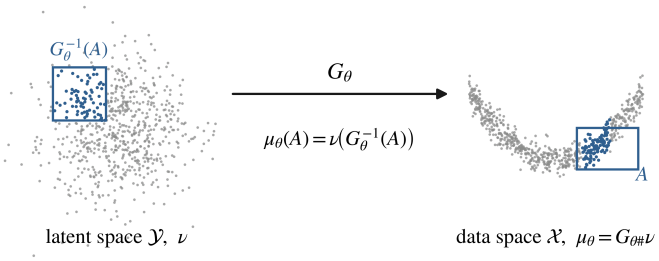


Figure 1.3.: What the definition says, drawn. The mass μ_θ assigns to a region A is by definition the mass ν assigns to the latent codes that land in A, so the preimage $G_\theta^{-1}(A)$ — a set, not an inverse map — is the only thing the construction needs.

Figure 1.3 is Equation 1.4 with the sets drawn in.

The following lemma is used repeatedly and is never stated in the lecture, so we prove it.

Lemma 1.1 (Change of variables)**.** *Let $T : \mathcal{X}_1 \to \mathcal{X}_2$ be measurable and let $h : \mathcal{X}_2 \to [0, \infty]$ be measurable. Then*

$$\int_{\mathcal{X}_2} h \, d(T_{\#}\mu) = \int_{\mathcal{X}_1} (h \circ T) \, d\mu. \tag{1.5}$$

Proof. Take first $h = \mathbf{1}_A$ for $A \in \mathcal{B}_2$. The left-hand side is $(T_{\#}\mu)(A)$, which is $\mu(T^{-1}(A))$ by Equation 1.4. The right-hand side is $\int \mathbf{1}_A(T(x)) \, d\mu(x)$, and $\mathbf{1}_A(T(x)) = \mathbf{1}_{T^{-1}(A)}(x)$, so it is also $\mu(T^{-1}(A))$. The two sides agree.

By linearity of the integral they agree for non-negative simple functions. For general measurable $h \geq 0$ choose simple $h_n \uparrow h$ pointwise; then $h_n \circ T \uparrow h \circ T$ pointwise, and monotone convergence applied on each side gives the result. An integrable h of arbitrary sign follows by splitting into h^+ and h^-. □

In particular, taking h to be a cost function turns an integral against a push-forward measure into an integral against the original one; this is the step that makes Monge's problem in Section 1.10 computable at all.

1.5. Metrics and divergences are not the same thing

Definition 1.3 (Metric)**.** A metric on a set $\mathcal{X}$ is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ such that for all $x, y, z \in \mathcal{X}$:

1. $d(x, y) \geq 0$;
2. $d(x, y) = 0$ if and only if $x = y$;
3. $d(x, y) = d(y, x)$;
4. $d(x, z) \leq d(x, y) + d(y, z)$.

Definition 1.4 (Divergence)**.** A divergence on $\mathcal{P}(\mathcal{X})$ is a function $D : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to [0, \infty]$ satisfying only conditions 1 and 2 of Definition 1.3. Symmetry and the triangle inequality are not required.

The distinction is exactly what separates the two halves of this chapter. The f-divergences of Section 1.6 satisfy only the first two conditions: $D_{\mathrm{KL}}(P\|Q)$ and $D_{\mathrm{KL}}(Q\|P)$ are different numbers, and there is no triangle inequality relating three distributions. The Wasserstein distance of Section 1.9 satisfies all four and is a metric in the full sense — we prove this as Theorem 1.3. Statistical distances thus split into a divergence branch, which leads to the GAN and f-GAN objectives, and a metric branch, which leads to W-GAN.

One more definition from the preliminaries is needed later, in the Kantorovich–Rubinstein theorem.

Definition 1.5 (Lipschitz continuity)**.** Given metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$, a function $f : \mathcal{X} \to \mathcal{Y}$ is Lipschitz continuous with constant $K \geq 0$ if $d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K\, d_{\mathcal{X}}(x_1, x_2)$ for all $x_1, x_2 \in \mathcal{X}$. We write $\mathrm{Lip}_1(\mathcal{X})$ for the real-valued functions on $\mathcal{X}$ that are Lipschitz with constant 1.

1.6. Divergences generated by a convex function

The first family of statistical distances compares P and Q through the ratio of their densities, passed through a convex function.

Definition 1.6 (f-divergence)**.** Let μ and ν be probability measures on a space Ω with $\mu \ll \nu$, and let f be a convex function with $f(1) = 0$. The f-divergence of μ from ν is

$$D_f(\mu\|\nu) := \int_\Omega f\left(\frac{d\mu}{d\nu}\right) d\nu. \tag{1.6}$$

If $\mu \ll \xi$ and $\nu \ll \xi$ for a common measure ξ on Ω, with densities $p = d\mu/d\xi$ and $q = d\nu/d\xi$, this reads

$$D_f(P\|Q) := \int_\Omega f\left(\frac{p(x)}{q(x)}\right) q(x)\, d\xi(x), \tag{1.7}$$

and in the discrete case, with P and Q probability mass functions,

$$D_f(P\|Q) := \sum_x Q(x)\, f\left(\frac{P(x)}{Q(x)}\right). \tag{1.8}$$

Two features of Equation 1.6 are worth naming before we go on, because both come back later. The integration is against the *second* argument, and the ratio is of the first to the second; swapping P and Q therefore gives a different number in general. And the definition presupposes $\mu \ll \nu$, so that $d\mu/d\nu$ exists at all by Theorem 1.1. When that fails, Equation 1.6 is not repaired by any pointwise convention: the standard extension splits μ into its absolutely continuous and singular parts and charges the singular part at the recession slope of f, which is Equation 1.16 below. Finiteness then depends on f at both ends, on $f(0)$ as well as on $\lim_{u\to\infty} f(u)/u$, and not on growth alone: reverse Kullback–Leibler has recession slope 0 and is still $+\infty$ on singular pairs, because $f(0) = +\infty$. This is the seed of Section 1.7.

The requirement $f(1) = 0$ is a normalisation, and convexity is what makes the whole family well behaved:

Proposition 1.1 (Non-negativity)**.** *For any convex f with $f(1) = 0$, $D_f(P\|Q) \geq 0$. If in addition f is strictly convex at 1, then $D_f(P\|Q) = 0$ if and only if $P = Q$.*

Proof. Write the divergence as an expectation under Q of the random variable $U = p(X)/q(X)$, with $X \sim Q$:

$$D_f(P\|Q) = \mathbb{E}_Q[f(U)].$$

Since f is convex, Jensen's inequality gives $\mathbb{E}_Q[f(U)] \geq f(\mathbb{E}_Q[U])$. The expectation inside is

$$\mathbb{E}_Q[U] = \int \frac{p(x)}{q(x)}\, q(x)\, d\xi(x) = \int p(x)\, d\xi(x) = 1,$$

because p is a probability density. Hence $D_f(P\|Q) \geq f(1) = 0$, which is the first claim.

For the second, suppose $D_f(P\|Q) = 0$. Convexity supplies a supporting line at $u = 1$: there is $a \in \mathbb{R}$ with $f(u) \geq f(1) + a(u-1) = a(u-1)$ for every $u > 0$. Since $\mathbb{E}_Q[U] = 1$, subtracting that line changes nothing:

$$0 = D_f(P\|Q) = \mathbb{E}_Q\big[f(U)\big] = \mathbb{E}_Q\big[f(U) - a(U-1)\big],$$

and the integrand is non-negative. A non-negative random variable with zero expectation vanishes almost surely, so $f(U) = a(U-1)$ Q-almost surely. If f is strictly convex at 1 the supporting line meets the graph only at $u = 1$, so $U = 1$ Q-almost surely; that is, $p = q$ almost everywhere and $P = Q$. Conversely if $P = Q$ then $U \equiv 1$ and the integrand is $f(1) = 0$ everywhere. □

Choosing f selects a member of the family.

Divergence	Generator $f(u)$	Symmetric	Bounded
Kullback–Leibler	$u \log u$	no	no
Reverse Kullback–Leibler	$-\log u$	no	no
Pearson χ^2	$(u-1)^2$	no	no
Squared Hellinger	$(\sqrt{u}-1)^2$	yes	yes, by 2
Total variation	$\frac{1}{2}\lvert u-1\rvert$	yes	yes, by 1
Jensen–Shannon	$(u+1)\log\frac{2}{u+1} + u\log u$	yes	yes, by $2\log 2$

1.6.1. Kullback–Leibler divergence

Take $f(t) = t\log t$, which is convex on $(0,\infty)$ and vanishes at $t = 1$. Substituting into Equation 1.8,

$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= \sum_x Q(x)\,\frac{P(x)}{Q(x)}\log\frac{P(x)}{Q(x)} \\ &= \sum_x P(x)\log\frac{P(x)}{Q(x)} \\ &= \sum_x P(x)\log P(x) \;-\; \sum_x P(x)\log Q(x) \\ &= H(P,Q) - H(P), \end{aligned} \tag{1.9}$$

where the cancellation in the second line is legitimate wherever $Q(x) > 0$, and the terms with $P(x) = 0$ contribute nothing under the convention $0\log 0 = 0$. The entropy and cross-entropy appearing in Equation 1.9 are

$$H(P) = -\sum_x P(x)\log P(x), \qquad H(P,Q) = -\sum_x P(x)\log Q(x). \tag{1.10}$$

Equation 1.9 is why D_{KL} is also called the relative entropy: it is the excess in the average number of nats needed to describe samples from P using a code built for Q, over the best code for P itself. It also explains why minimising a cross-entropy loss over θ and minimising $D_{\mathrm{KL}}(P\|Q_\theta)$ are the same problem — $H(P)$ does not depend on θ.

Note where Equation 1.9 breaks. If there is a point with $P(x) > 0$ and $Q(x) = 0$ then the ratio is infinite and $D_{\mathrm{KL}}(P\|Q) = +\infty$: the generator $f(t) = t\log t$ satisfies $f(t)/t \to \infty$. So D_{KL} is finite only when $P \ll Q$, which is precisely the hypothesis of Definition 1.6 and, as we will see, precisely what a generative model cannot guarantee.

Taking $f(u) = -\log u$ instead gives

$$D_f(P\|Q) = \sum_x Q(x)\log\frac{Q(x)}{P(x)} = D_{\mathrm{KL}}(Q\|P), \tag{1.11}$$

the reverse Kullback–Leibler divergence, finite only when $Q \ll P$. The asymmetry is not a technicality: minimising $D_{\mathrm{KL}}(P\|Q_\theta)$ over θ penalises putting no mass where P has some, so the fitted model spreads out to cover every mode; minimising $D_{\mathrm{KL}}(Q_\theta\|P)$ penalises putting mass where P has none, so the fitted model contracts onto one mode and ignores the rest. Figure 1.4 shows the two fits side by side.

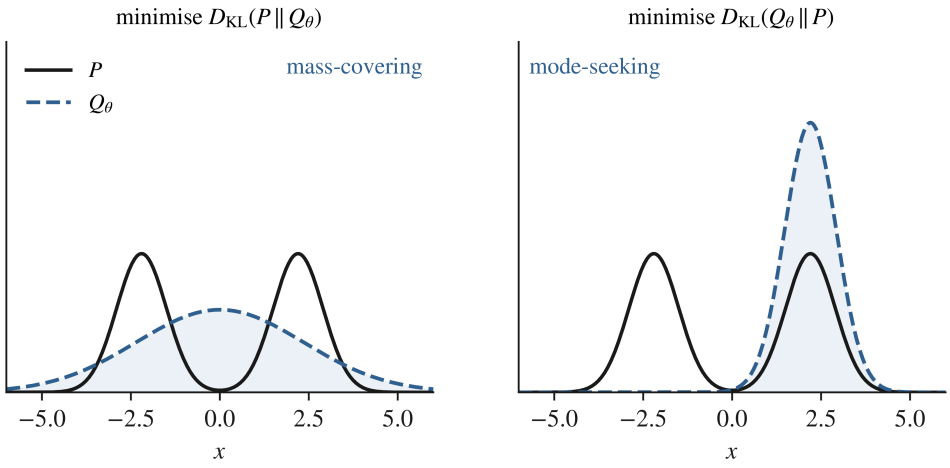


Figure 1.4.: Mass-covering versus mode-seeking. A unimodal Q_θ is fitted to a bimodal P under the two orderings of the Kullback–Leibler divergence. Minimising $D_{\mathrm{KL}}(P\|Q_\theta)$ forces Q_θ to cover both modes and so to place mass in the valley between them; minimising $D_{\mathrm{KL}}(Q_\theta\|P)$ lets it lock onto a single mode. This is the divergence-level description of mode collapse.

1.6.2. Total variation

The generator $f(u) = \frac{1}{2}|u-1|$ is convex and vanishes at $u=1$, so it defines an f-divergence. It is the one member of the family that is also a metric, and the one that will reappear in Section 1.8 from the other direction entirely.

Proposition 1.2 (Total variation)**.** *With $f(u) = \frac{1}{2}|u-1|$,*

$$D_f(P\|Q) = \tfrac{1}{2}\int |p(x)-q(x)|\,d\xi(x) = \sup_A |P(A)-Q(A)| =: \delta(P,Q), \tag{1.12}$$

the supremum being over measurable sets A.

Proof. The first equality is immediate from Equation 1.7:

$$\int \tfrac{1}{2}\left|\frac{p}{q}-1\right| q\,d\xi = \tfrac{1}{2}\int |p-q|\,d\xi.$$

For the second, write $r = p-q$ and let $A^* = \{x : p(x) > q(x)\}$. Since P and Q are probability measures, $\int r\,d\xi = 0$, so

$$\int_{A^*} r\,d\xi = -\int_{(A^*)^c} r\,d\xi = \int_{(A^*)^c} |r|\,d\xi.$$

The integrand $|r|$ equals r on A^* and $-r$ off it, so

$$\int |r|\,d\xi = \int_{A^*} r\,d\xi + \int_{(A^*)^c} |r|\,d\xi = 2\int_{A^*} r\,d\xi,$$

giving $P(A^*)-Q(A^*) = \frac{1}{2}\int|r|\,d\xi$. That attains the supremum. For the upper bound, any measurable A satisfies

$$|P(A)-Q(A)| = \left|\int_A r\,d\xi\right| \le \max\left\{\int_A r^+ d\xi, \int_A r^- d\xi\right\} \le \int r^+ d\xi = \tfrac{1}{2}\int |r|\,d\xi,$$

where the last equality uses $\int r^+ d\xi = \int r^- d\xi$, itself a restatement of $\int r\,d\xi = 0$. □

Unlike the Kullback–Leibler divergence, Equation 1.12 never exceeds 1 and never needs absolute continuity: the generator $\frac{1}{2}|u-1|$ grows linearly, so $f(u)/u$ stays bounded. Boundedness will turn out to be exactly the problem rather than the cure.

1.6.3. Jensen–Shannon divergence

The Jensen–Shannon divergence symmetrises the Kullback–Leibler divergence by comparing both arguments against their average.

Definition 1.7 (Jensen–Shannon divergence)**.** For probability measures P and Q, set $M = \frac{1}{2}(P+Q)$. Then

$$D_{\mathrm{JS}}(P\|Q) := \tfrac{1}{2}D_{\mathrm{KL}}(P\|M) + \tfrac{1}{2}D_{\mathrm{KL}}(Q\|M). \tag{1.13}$$

Symmetry is immediate, since M is unchanged by swapping P and Q. The substantive point is that Equation 1.13 is always finite, which the Kullback–Leibler divergence is not.

Lemma 1.2 (The mixture dominates both arguments)**.** *$P \ll M$ and $Q \ll M$, so both Radon–Nikodym derivatives dP/dM and dQ/dM exist and Equation 1.13 is well defined for every pair P, Q.*

Proof. Let A be measurable with $M(A) = 0$. By definition of M,

$$0 = M(A) = \tfrac{1}{2}P(A) + \tfrac{1}{2}Q(A).$$

Both $P(A) \geq 0$ and $Q(A) \geq 0$, and a sum of non-negative terms vanishes only if each vanishes, so $P(A) = Q(A) = 0$. By Definition 1.1 this is exactly $P \ll M$ and $Q \ll M$. □

This is the whole advantage over the Kullback–Leibler divergence. To use $D_{\mathrm{KL}}(P\|Q)$ we must know in advance that $P \ll Q$, which for a model distribution and an unknown data distribution we never do.

Proposition 1.3 (Jensen–Shannon is bounded)**.** $0 \leq D_{\mathrm{JS}}(P\|Q) \leq \log 2$*, with the upper bound attained exactly when P and Q are mutually singular.*

Proof. The lower bound is Proposition 1.1. For the upper bound, note that pointwise $M = \frac{1}{2}(P+Q) \geq \frac{1}{2}P$ as measures, so $dP/dM \leq 2$ M-almost everywhere. Hence

$$D_{\mathrm{KL}}(P\|M) = \int \log\!\left(\frac{dP}{dM}\right) dP \leq \int \log 2 \, dP = \log 2,$$

and the same bound holds for $D_{\mathrm{KL}}(Q\|M)$. Substituting both into Equation 1.13 gives $D_{\mathrm{JS}} \leq \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = \log 2$.

Equality requires $dP/dM = 2$ P-almost everywhere. Because $P + Q = 2M$, the two derivatives satisfy $dP/dM + dQ/dM = 2$ M-almost everywhere, so on the set $S = \{dP/dM = 2\}$ we have $dQ/dM = 0$ and therefore

$$Q(S) = \int_S \frac{dQ}{dM}\, dM = 0,$$

while $P(S) = 1$. So P and Q are carried by the disjoint sets S and S^c, which is mutual singularity. Conversely, if P and Q live on disjoint sets S and S^c, then on S we have $M = \frac{1}{2}P$ and $dP/dM = 2$, and symmetrically on S^c, so both divergences equal $\log 2$. □

Finally we connect Equation 1.13 back to Definition 1.6, and here the lecture's two slides do not quite agree with each other.

Proposition 1.4 (The Jensen–Shannon generator carries a factor of two)**.** *Let* $f_{\mathrm{JS}}(u) = (u+1)\log\frac{2}{u+1} + u\log u$. *Then* f_{JS} *is convex with* $f_{\mathrm{JS}}(1) = 0$, *and*

$$D_{f_{\mathrm{JS}}}(P\|Q) = 2\, D_{\mathrm{JS}}(P\|Q). \tag{1.14}$$

Proof. That $f_{\mathrm{JS}}(1) = 2\log 1 + 1\cdot\log 1 = 0$ is direct. For convexity, differentiate twice on $u > 0$:

$$f'_{\mathrm{JS}}(u) = \log\frac{2}{u+1} - 1 + \log u + 1 = \log\frac{2u}{u+1}, \qquad f''_{\mathrm{JS}}(u) = \frac{1}{u} - \frac{1}{u+1} = \frac{1}{u(u+1)} > 0.$$

For the identity, substitute into Equation 1.8 with $u = P(x)/Q(x)$ and clear denominators:

$$\begin{aligned} D_{f_{\mathrm{JS}}}(P\|Q) &= \sum_x Q\left[\left(\frac{P}{Q}+1\right)\log\frac{2}{\frac{P}{Q}+1} + \frac{P}{Q}\log\frac{P}{Q}\right] \\ &= \sum_x \left[(P+Q)\log\frac{2Q}{P+Q} + P\log\frac{P}{Q}\right]. \end{aligned}$$

Now $M = \frac{1}{2}(P+Q)$ gives $\dfrac{2Q}{P+Q} = \dfrac{Q}{M}$, so

$$
\begin{aligned}
D_{f_{\mathrm{JS}}}(P\|Q) &= \sum_x \left[P \log \frac{Q}{M} + Q \log \frac{Q}{M} + P \log \frac{P}{Q} \right] \\
&= \sum_x \left[P \log\!\left(\frac{Q}{M} \cdot \frac{P}{Q} \right) + Q \log \frac{Q}{M} \right] \\
&= \sum_x P \log \frac{P}{M} + \sum_x Q \log \frac{Q}{M} \\
&= D_{\mathrm{KL}}(P\|M) + D_{\mathrm{KL}}(Q\|M),
\end{aligned}
$$

which by Equation 1.13 is $2D_{\mathrm{JS}}(P\|Q)$. □

The factor of two is not an error in either statement; it is a collision of two conventions. Equation 1.13 is the information-theoretic normalisation, under which the divergence is bounded by $\log 2$ and equals the mutual information between a sample and a fair coin choosing which of P and Q produced it. f_{JS} is the generator tabulated in the f-GAN literature, which omits the $\frac{1}{2}$ so that the associated variational objective matches the original GAN loss up to an additive $\log 4$. **This chapter uses Equation 1.13 throughout**, so that the bound of Proposition 1.3 reads $\log 2$; a reader comparing against a table of f-divergence generators should expect the other normalisation and divide by two.

1.7. Disjoint supports and the vanishing gradient

Everything so far has quietly assumed that the density ratio p/q means something. This section shows what happens when it does not, which for generative models is the typical case rather than a pathology. The conclusion is that every f-divergence saturates at a constant that does not depend on how far apart the two distributions are, and that a constant has no gradient.

To say this cleanly we need the standard extension of Definition 1.6 to pairs that are not absolutely continuous. Let $P = P^{\mathrm{ac}} + P^{\perp}$ be the Lebesgue decomposition of P with respect to Q, and define the recession slope

$$
f'(\infty) := \lim_{u \to \infty} \frac{f(u)}{u} \in (-\infty, +\infty], \tag{1.15}
$$

which exists in the extended reals because f is convex. The f-divergence is then

$$
D_f(P\|Q) := \int f\!\left(\frac{dP^{\mathrm{ac}}}{dQ} \right) dQ \;+\; f'(\infty)\, P^{\perp}(\Omega), \tag{1.16}
$$

which agrees with Equation 1.6 when $P \ll Q$, since then $P^{\perp} = 0$.

Theorem 1.2 (Saturation on disjoint supports)**.** *Let P and Q be mutually singular probability measures, written $P \perp Q$. Then for every generator f,*

$$D_f(P\|Q) = f(0) + f'(\infty), \tag{1.17}$$

a constant depending only on f and not on P or Q.

Proof. Mutual singularity means there is a measurable set S with $P(S) = 1$ and $Q(S) = 0$. Then P is carried by S and Q by S^c, so the Lebesgue decomposition of P with respect to Q is $P^{\mathrm{ac}} = 0$ and $P^{\perp} = P$, whence $dP^{\mathrm{ac}}/dQ = 0$ Q-almost everywhere. Substituting into Equation 1.16,

$$D_f(P\|Q) = \int f(0)\, dQ + f'(\infty)\, P(\Omega) = f(0) \cdot 1 + f'(\infty) \cdot 1,$$

since P and Q are probability measures. □

Evaluating Equation 1.17 for the generators tabulated earlier gives the saturation values directly, and they agree with the bounds proved above.

Divergence	$f(0)$	$f'(\infty)$	Value when $P \perp Q$
Kullback–Leibler	0	$+\infty$	$+\infty$
Reverse Kullback–Leibler	$+\infty$	0	$+\infty$
Pearson χ^2	1	$+\infty$	$+\infty$
Squared Hellinger	1	1	2
Total variation	$\frac{1}{2}$	$\frac{1}{2}$	1
Jensen–Shannon (as f_{JS})	$\log 2$	$\log 2$	$2 \log 2$

The last row is a useful consistency check on Proposition 1.4: dividing the saturation value $2 \log 2$ by the factor of two recovers exactly the bound $\log 2$ of Proposition 1.3, attained precisely on mutually singular pairs. The Pearson χ^2 row is the one generator the chapter tabulates but never works through; Section 1.15.1 does so.

1.7.1. What this costs a generator

Example 1.1 (Two parallel segments)**.** Let $Z \sim \mathrm{Uniform}[0,1]$ and work in $\mathbb{R}^2$. Define μ_0 to be the law of $(0, Z)$ and, for $\theta \in \mathbb{R}$, define μ_θ to be the law of (θ, Z). Each is uniform on a vertical unit segment, and for $\theta \neq 0$ the two segments are disjoint. Then

$$W_1(\mu_0, \mu_\theta) = |\theta|, \qquad D_{\mathrm{JS}}(\mu_0 \| \mu_\theta) = \begin{cases} \log 2, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases},$$

$$\delta(\mu_0, \mu_\theta) = \begin{cases} 1, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}, \qquad D_{\mathrm{KL}}(\mu_0 \| \mu_\theta) = \begin{cases} +\infty, & \theta \neq 0 \\ 0, & \theta = 0. \end{cases}$$

Proof. For $\theta \neq 0$ the supports are the disjoint segments $\{0\} \times [0,1]$ and $\{\theta\} \times [0,1]$, so $\mu_0 \perp \mu_\theta$ and the last three claims are Theorem 1.2 read off the table, with the Jensen–Shannon entry divided by two to match the normalisation of Equation 1.13. At $\theta = 0$ the two measures coincide and every divergence vanishes by Proposition 1.1.

For W_1, whose definition is Definition 1.10 below, take the coupling π induced by the map $(0, z) \mapsto (\theta, z)$, that is, the law of $((0, Z), (\theta, Z))$. Its marginals are μ_0 and μ_θ by construction, and its cost is

$$\mathbb{E}\,\|(0, Z) - (\theta, Z)\| = \mathbb{E}|\theta| = |\theta|,$$

so $W_1(\mu_0, \mu_\theta) \leq |\theta|$. Conversely, let π be any coupling and let $(X, Y) \sim \pi$. Writing X_1, Y_1 for first coordinates, $X_1 = 0$ almost surely and $Y_1 = \theta$ almost surely, so $\|X - Y\| \geq |X_1 - Y_1| = |\theta|$ pointwise, and taking expectations gives $\mathbb{E}\|X - Y\| \geq |\theta|$. Hence $W_1(\mu_0, \mu_\theta) \geq |\theta|$, and the two bounds agree. □

Figure 1.5 is the picture to keep. Suppose we train a generator whose only parameter is the horizontal offset θ, and we descend the divergence in θ. Under W_1 the objective is $|\theta|$, with derivative $\mathrm{sign}(\theta)$: from any starting point the gradient points at the answer. Under the Jensen–Shannon divergence the objective is the constant $\log 2$ for every $\theta \neq 0$, so its derivative is zero everywhere except at the single point we are trying to find. The optimiser receives no information at all. Under the Kullback–Leibler divergence the objective is not even finite.

This is not an artefact of a one-dimensional toy. By Equation 1.1 the model distribution μ_θ is supported on $G_\theta(\mathcal{Y})$, the image of a latent space of dimension perhaps 100, embedded in a pixel space of dimension perhaps 10^5; and the data are taken to lie on a manifold of comparably small dimension. Two sets of dimension 100 inside $\mathbb{R}^{10^5}$ that are not deliberately arranged to meet will be mutually singular, so Theorem 1.2 applies at essentially every θ, not just at isolated ones. A generator trained against an f-divergence

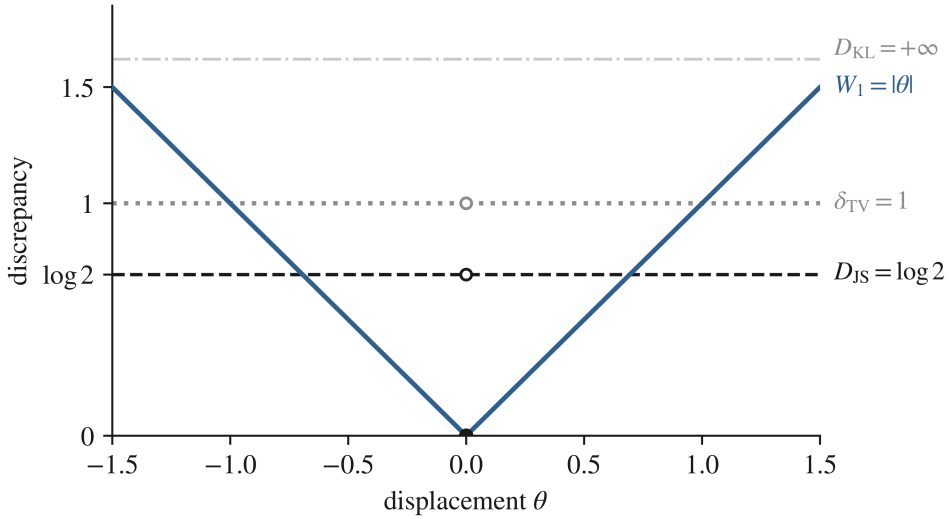


Figure 1.5.: The four discrepancies of Example 1.1 as functions of the displacement θ between two parallel unit segments in the plane. The Wasserstein distance is the only one that varies with θ; the Jensen–Shannon divergence and the total variation jump to their saturation values the instant the supports separate and stay there, and the Kullback–Leibler divergence is infinite throughout. A gradient step on any curve but the first has nothing to follow.

is therefore relying on the discriminator never becoming very good — an unstable thing to rely on, and the observed behaviour of the original GAN.

Two escapes exist. One is to stop comparing densities pointwise and start comparing expectations of test functions, which is Section 1.8. The other is to measure how far mass must physically move, which is Section 1.10 onwards. They turn out to be the same escape, and they meet in Theorem 1.7.

1.8. Integral probability metrics

An f-divergence asks how the two distributions differ at each point. An integral probability metric asks instead how differently they answer a restricted set of questions.

Definition 1.8 (Integral probability metric)**.** Let $\mathcal{F}$ be a class of real measurable functions on $\mathcal{X}$. The integral probability metric associated with $\mathcal{F}$ is

$$d_{\mathcal{F}}(\mu, \nu) := \sup_{h \in \mathcal{F}} \left| \int_{\mathcal{X}} h \, d\mu - \int_{\mathcal{X}} h \, d\nu \right|. \tag{1.18}$$

The name "dual-norm view" comes from reading Equation 1.18 as a norm on the signed measure $\mu - \nu$. Probability measures pair with bounded functions by integration, so $\mu - \nu$ acts as a linear functional on the space of test functions; Equation 1.18 is the operator norm of that functional, measured over the unit ball $\mathcal{F}$. Different choices of $\mathcal{F}$ are different norms on the same object, and, as we are about to see, three familiar quantities are three choices. How literally that dual-norm reading may be taken, and which topology each choice induces, are examined in Section 1.15.2.

Throughout we take $\mathcal{F}$ symmetric, $\mathcal{F} = -\mathcal{F}$, as all three classes below are; the absolute value in Equation 1.18 may then be dropped, since h and $-h$ both compete in the supremum.

Proposition 1.5 (Integral probability metrics are pseudo-metrics)**.** *Let every $h \in \mathcal{F}$ be integrable with respect to both measures. Then $d_{\mathcal{F}}$ is non-negative, symmetric, vanishes when $\mu = \nu$, and satisfies the triangle inequality. It is a metric on $\mathcal{P}(\mathcal{X})$ if and only if $\mathcal{F}$ is measure-determining, meaning that $\int h\, d\mu = \int h\, d\nu$ for all $h \in \mathcal{F}$ implies $\mu = \nu$.*

Proof. Non-negativity is clear, as is $d_{\mathcal{F}}(\mu, \mu) = 0$. Symmetry holds because the absolute value is unchanged when μ and ν are exchanged. For the triangle inequality, fix $h \in \mathcal{F}$ and any third measure ρ; then

$$\left|\int h\, d\mu - \int h\, d\nu\right| \le \left|\int h\, d\mu - \int h\, d\rho\right| + \left|\int h\, d\rho - \int h\, d\nu\right| \le d_{\mathcal{F}}(\mu, \rho) + d_{\mathcal{F}}(\rho, \nu).$$

The right-hand side does not depend on h, so taking the supremum over $h \in \mathcal{F}$ on the left preserves the inequality.

Finally, $d_{\mathcal{F}}(\mu, \nu) = 0$ says exactly that $\int h\, d\mu = \int h\, d\nu$ for every $h \in \mathcal{F}$. That this forces $\mu = \nu$ is the definition of measure-determining, so the second axiom of Definition 1.3 holds precisely under that hypothesis. □

Note how little was needed: no density ratio, no absolute continuity, and no common dominating measure. This is why integral probability metrics survive the situation that defeats Section 1.7.

1.8.1. Three instances

Total variation. Take $\mathcal{F}$ to be the indicators $\{\mathbf{1}_A : A \text{ measurable}\}$. Then $\int \mathbf{1}_A d\mu - \int \mathbf{1}_A d\nu = \mu(A) - \nu(A)$, so Equation 1.18 reduces to $\sup_A |\mu(A) - \nu(A)|$, which is $\delta(\mu, \nu)$ by Proposition 1.2. Taking instead the sup-norm ball $\{h : \|h\|_\infty \le 1\}$ gives $2\delta(\mu, \nu)$, since the supremum is attained at $h = \operatorname{sign}(p - q)$.

So total variation belongs to both families. That is not a coincidence about our proof, and it is essentially the only overlap: up to a constant multiple, total variation is the unique non-trivial statistical distance that is both an f-divergence and an integral probability metric, as Figure 1.6 records.

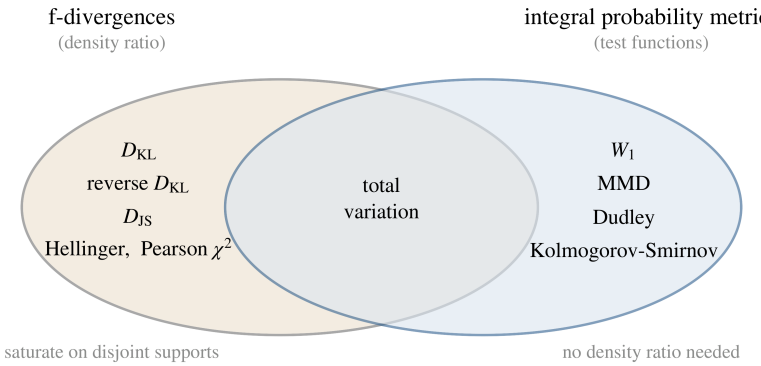


Figure 1.6.: The two families of statistical distances and what lies in each. Every f-divergence saturates on disjoint supports by Theorem 1.2; the integral probability metrics on the right need no density ratio and need not saturate. Total variation is essentially the only member of both.

Wasserstein-1. Take $\mathcal{F} = \mathrm{Lip}_1(\mathcal{X})$ from Definition 1.5. The resulting integral probability metric is W_1. This is the Kantorovich–Rubinstein theorem and it is genuinely a theorem rather than a definition, because W_1 is defined in Definition 1.10 as an infimum over couplings with no test functions in sight. We prove it as Theorem 1.7, once duality is available.

Maximum mean discrepancy. Let k be a positive-definite kernel on $\mathcal{X}$ and $\mathcal{H}_k$ its reproducing kernel Hilbert space, so that $h(x) = \langle h, k(\cdot, x)\rangle_{\mathcal{H}_k}$ for every $h \in \mathcal{H}_k$. Take $\mathcal{F}$ to be the unit ball $\{h : \|h\|_{\mathcal{H}_k} \le 1\}$.

Proposition 1.6 (Maximum mean discrepancy)**.** *Assume* $\int \sqrt{k(x,x)}\, d\mu(x)$ *and* $\int \sqrt{k(y,y)}\, d\nu(y)$ *are finite, and define the mean embeddings* $m_\mu = \int k(\cdot, x)\, d\mu(x)$ *and* $m_\nu = \int k(\cdot, y)\, d\nu(y)$ *in* $\mathcal{H}_k$*. Then*

$$\mathrm{MMD}(\mu, \nu) := d_{\mathcal{F}}(\mu, \nu) = \|m_\mu - m_\nu\|_{\mathcal{H}_k}, \tag{1.19}$$

and

$$\mathrm{MMD}^2(\mu, \nu) = \mathbb{E}_{x,x' \sim \mu} k(x, x') - 2\, \mathbb{E}_{x \sim \mu,\, y \sim \nu} k(x, y) + \mathbb{E}_{y,y' \sim \nu} k(y, y'). \tag{1.20}$$

Proof. By the reproducing property and the finiteness assumption, integration against μ is an inner product with the mean embedding:

$$\int h \, d\mu = \int \langle h, k(\cdot, x) \rangle_{\mathcal{H}_k} \, d\mu(x) = \left\langle h, \int k(\cdot, x) \, d\mu(x) \right\rangle_{\mathcal{H}_k} = \langle h, m_\mu \rangle_{\mathcal{H}_k},$$

the exchange of integral and inner product being justified by the same finiteness assumption. Subtracting the same identity for ν,

$$\int h \, d\mu - \int h \, d\nu = \langle h, m_\mu - m_\nu \rangle_{\mathcal{H}_k}.$$

Taking the supremum of the absolute value over the unit ball is the definition of the dual norm in a Hilbert space, which by Cauchy–Schwarz equals $\|m_\mu - m_\nu\|_{\mathcal{H}_k}$, attained at $h = (m_\mu - m_\nu)/\|m_\mu - m_\nu\|$. That is Equation 1.19.

For Equation 1.20, expand the squared norm and use the reproducing property once more:

$$\langle m_\mu, m_\nu \rangle_{\mathcal{H}_k} = \left\langle \int k(\cdot, x) d\mu(x), \int k(\cdot, y) d\nu(y) \right\rangle = \iint \langle k(\cdot, x), k(\cdot, y) \rangle \, d\mu \, d\nu = \iint k(x, y) \, d\mu \, d\nu,$$

and similarly for $\langle m_\mu, m_\mu \rangle$ and $\langle m_\nu, m_\nu \rangle$. Substituting into $\|m_\mu - m_\nu\|^2 = \langle m_\mu, m_\mu \rangle - 2\langle m_\mu, m_\nu \rangle + \langle m_\nu, m_\nu \rangle$ gives the result. □

Equation 1.20 is what makes the maximum mean discrepancy practical: it is written entirely in terms of expectations of the kernel, so it can be estimated from two samples without any density estimate and without solving the supremum. The two standard estimators, and the condition on the kernel under which the maximum mean discrepancy separates distributions at all, are in Section 1.15.3.

This is the discrepancy behind the moment-matching generative models, and it is where McGAN and MMD-GAN enter in the next lecture.

1.9. Wasserstein distances

We now build the second family. Instead of comparing μ and ν where they sit, we ask what it would cost to carry the mass of one onto the other. The bookkeeping device for "carrying mass" is a joint distribution.

Definition 1.9 (Coupling)**.** Let $\mu \in \mathcal{P}(\mathcal{X})$ and $\nu \in \mathcal{P}(\mathcal{Y})$. A coupling of μ and ν is a probability measure π on $\mathcal{X} \times \mathcal{Y}$ whose marginals are μ and ν:

$$\pi(A \times \mathcal{Y}) = \mu(A), \qquad \pi(\mathcal{X} \times B) = \nu(B) \tag{1.21}$$

for all measurable $A \subseteq \mathcal{X}$, $B \subseteq \mathcal{Y}$. The set of couplings is written $\Pi(\mu, \nu)$.

Read $\pi(dx, dy)$ as the amount of mass moved from x to y. The two constraints in Equation 1.21 say that everything sent out of x adds up to what μ had there, and everything arriving at y adds up to what ν needs there. The set $\Pi(\mu, \nu)$ is never empty: the product measure $\mu \otimes \nu$, which ignores where mass came from, is always a coupling.

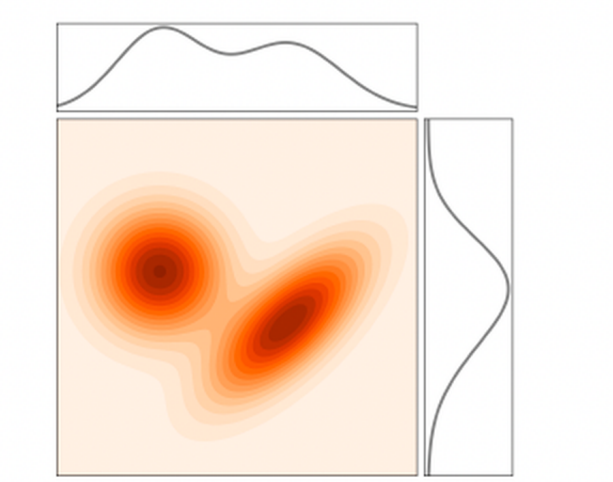


Figure 1.7.: A coupling and the two marginals it must reproduce. The shaded density is a joint π on the plane; the curve above is its first marginal and the curve to the right its second, each obtained by integrating out the other coordinate. Equation 1.21 asks exactly that these two curves be μ and ν. Nothing asks the joint to be the product of them — this one is not, having two modes where a product would have four — and the freedom left over is the set $\Pi(\mu, \nu)$ that Definition 1.10 minimises over. From the course slides.

Definition 1.10 (Wasserstein distance)**.** Let $(\mathcal{M}, d)$ be a Polish metric space and let $p \geq 1$. For $\mu, \nu \in \mathcal{P}(\mathcal{M})$ with finite p-th moments, the p-th Wasserstein distance is

$$W_p(\mu, \nu) := \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{M} \times \mathcal{M}} d(x, y)^p \, d\pi(x, y) \right)^{1/p} = \left(\inf_{\pi \in \Pi(\mu,\nu)} \mathbb{E}_\pi \big[d(X, Y)^p \big] \right)^{1/p}. \tag{1.22}$$

For $p = 1$ this is the earth-mover distance: the least total amount of "mass times distance" needed to rearrange one pile of earth into the other. The name is worth taking literally, because it is the source of the intuition that Figure 1.5 makes precise — moving two separated piles closer together reduces the cost continuously, whereas asking whether they overlap gives the same answer until the instant they touch.

That Equation 1.22 really is a metric, and not merely a divergence, is the structural fact that distinguishes this family from Section 1.6.

Theorem 1.3 (Wasserstein is a metric)**.** *For $p \geq 1$, W_p satisfies all four axioms of Definition 1.3 on the set of probability measures on $\mathcal{M}$ with finite p-th moments.*

Proof. Non-negativity is immediate, since $d^p \geq 0$.

Symmetry. If $\pi \in \Pi(\mu, \nu)$, let $S(x, y) = (y, x)$ and put $\pi' = S_{\#}\pi$. By Equation 1.4 its marginals are those of π in the opposite order, so $\pi' \in \Pi(\nu, \mu)$, and by Lemma 1.1 its cost is $\int d(y, x)^p d\pi = \int d(x, y)^p d\pi$ because d is symmetric. So $S_{\#}$ is a cost-preserving bijection between $\Pi(\mu, \nu)$ and $\Pi(\nu, \mu)$, and the two infima agree.

Identity of indiscernibles. If $\mu = \nu$, the coupling $\pi = (\mathrm{id}, \mathrm{id})_{\#}\mu$ concentrated on the diagonal has cost $\int d(x, x)^p d\mu = 0$, so $W_p(\mu, \mu) = 0$. Conversely suppose $W_p(\mu, \nu) = 0$ and let π be an optimal coupling, which exists by Theorem 1.5 — this is the one place the proof uses the standing assumption of Definition 1.10 that $\mathcal{M}$ is Polish, the cost d^p being continuous and hence lower semicontinuous. Then $\int d(x, y)^p d\pi = 0$ with a non-negative integrand, so $d(x, y) = 0$ and hence $x = y$ for π-almost every pair; that is, π is concentrated on the diagonal $\Delta = \{(x, x)\}$, a measurable set since $\mathcal{M}$ is metric. As $\pi(\Delta) = 1$, intersecting with Δ leaves any π-measure unchanged, and for measurable A the sets $(A \times \mathcal{M}) \cap \Delta$ and $(\mathcal{M} \times A) \cap \Delta$ coincide — both are $\{(x, x) : x \in A\}$. Hence

$$\mu(A) = \pi(A \times \mathcal{M}) = \pi(\mathcal{M} \times A) = \nu(A),$$

and $\mu = \nu$.

Triangle inequality. Let μ_1, μ_2, μ_3 be given and take $\pi_{12} \in \Pi(\mu_1, \mu_2)$ and $\pi_{23} \in \Pi(\mu_2, \mu_3)$. The gluing lemma provides a measure σ on $\mathcal{M} \times \mathcal{M} \times \mathcal{M}$ whose projection onto the first two coordinates is π_{12} and onto the last two is π_{23}; it is built by disintegrating both couplings over their shared marginal μ_2 and setting $d\sigma(x_1, x_2, x_3) = d\pi_{12}(x_1 \mid x_2)\, d\pi_{23}(x_3 \mid x_2)\, d\mu_2(x_2)$. Its projection onto the first and third coordinates is some $\pi_{13} \in \Pi(\mu_1, \mu_3)$. Then

$$
\begin{aligned}
W_p(\mu_1,\mu_3) &\le \left(\int d(x_1,x_3)^p \, d\sigma\right)^{1/p} \\
&\le \left(\int (d(x_1,x_2)+d(x_2,x_3))^p d\sigma\right)^{1/p} \\
&\le \left(\int d(x_1,x_2)^p d\sigma\right)^{1/p} + \left(\int d(x_2,x_3)^p d\sigma\right)^{1/p} \\
&= \left(\int d^p \, d\pi_{12}\right)^{1/p} + \left(\int d^p \, d\pi_{23}\right)^{1/p},
\end{aligned}
$$

using the triangle inequality for d in the second step and Minkowski's inequality in $L^p(\sigma)$ in the third. Taking the infimum over π_{12} and π_{23} separately gives $W_p(\mu_1,\mu_3) \le W_p(\mu_1,\mu_2) + W_p(\mu_2,\mu_3)$. □

By Jensen's inequality $W_1 \le W_p$ for $p \ge 1$, so the Wasserstein distances form an increasing family. We will use W_1 almost exclusively, because it is the one with a dual formulation simple enough to implement. Being a metric is not by itself what matters for Section 1.14 — *which* topology the metric induces is — and Section 1.15.4 settles that.

1.10. Monge's formulation

Historically the transport problem was posed not with couplings but with maps: each grain of earth is assigned one destination.

Definition 1.11 (Transportation map)**.** A measurable $T : \mathcal{X} \to \mathcal{Y}$ transports $\mu \in \mathcal{P}(\mathcal{X})$ to $\nu \in \mathcal{P}(\mathcal{Y})$ if

$$
\nu(B) = \mu(T^{-1}(B)) \quad \text{for all measurable } B, \tag{1.23}
$$

that is, if $\nu = T_{\#}\mu$ in the sense of Definition 1.2.

Figure 1.8 also shows what will go wrong. A map sends each source point somewhere; it cannot send one point to two places. When ν has mass where no single destination will do — the two humps on the right, fed from one hump on the left — a transport map may fail to exist at all, which is Proposition 1.7 below.

Definition 1.12 (Monge's problem)**.** Given a cost $c : \mathcal{X} \times \mathcal{Y} \to [0,\infty]$, Monge's optimal transport problem is

$$
\min_T \ \mathbb{M}(T) := \int_{\mathcal{X}} c(x, T(x)) \, d\mu(x) \qquad \text{subject to} \quad \nu = T_{\#}\mu. \tag{1.24}
$$

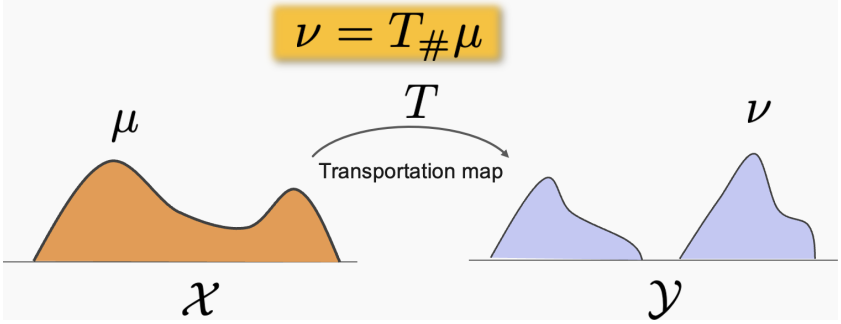


Figure 1.8.: A transportation map. Every bit of mass under μ on $\mathcal{X}$ is assigned a destination by T, and the mass that arrives is ν on $\mathcal{Y}$. The constraint Equation 1.23 is the whole of Monge's requirement: nothing is said about how T is built, only about where the mass ends up. From the course slides (Ye 2026a, 36).

The objective is benign; the constraint is not. The map $T \mapsto T_{\#}\mu$ is nonlinear, and the feasible set $\{T : T_{\#}\mu = \nu\}$ is not convex: if T_0 and T_1 both push μ to ν, there is no reason for $\frac{1}{2}(T_0 + T_1)$ to do so. So Equation 1.24 cannot be attacked by the standard machinery of convex optimisation. Worse, it can be infeasible.

Proposition 1.7 (Monge's problem may admit no solution)**.** *Let $\mu = \delta_0$ on $\mathbb{R}$ and $\nu = \frac{1}{2}(\delta_{-1} + \delta_1)$. Then no measurable T satisfies $T_{\#}\mu = \nu$, so Equation 1.24 has no feasible point at all.*

Proof. Suppose $T_{\#}\mu = \nu$. For any measurable B, Equation 1.23 gives $\nu(B) = \mu(T^{-1}(B)) = \delta_0(T^{-1}(B))$, which is 1 if $0 \in T^{-1}(B)$ and 0 otherwise. So $\nu(B) \in \{0, 1\}$ for every B. But $\nu(\{1\}) = \frac{1}{2}$, a contradiction. □

The obstruction is plain once seen: a map sends the single atom at 0 somewhere, and it can only send it to one place, whereas ν requires the mass to end up in two. A map cannot split mass. This is exactly what Section 1.11 repairs. Atoms are the crudest obstruction but not the only one; Section 1.15.5 gives two more, one of which shows that a non-atomic μ does not make Monge's problem well behaved either.

When a solution does exist, the one-dimensional case can be written down.

Theorem 1.4 (Transport on the line)**.** *Let μ, ν be probability measures on $\mathbb{R}$ with continuous, strictly increasing distribution functions F and G. Then $T = G^{-1} \circ F$ satisfies $T_{\#}\mu = \nu$, and for $c(x, y) = |x - y|^p$ with $p \geq 1$ it is optimal, giving*

$$W_p(\mu, \nu) = \left(\int_0^1 \left| F^{-1}(z) - G^{-1}(z) \right|^p dz \right)^{1/p} = \left(\int_{\mathbb{R}} \left| x - G^{-1}(F(x)) \right|^p dF(x) \right)^{1/p}. \quad (1.25)$$

Proof. We prove the push-forward claim and the formula; optimality of the monotone map for convex costs is classical and we cite it rather than reprove it.

Fix $y \in \mathbb{R}$. Since G is continuous and strictly increasing, and likewise F,

$$\begin{aligned}(T_{\#}\mu)((-\infty, y]) &= \mu(\{x : G^{-1}(F(x)) \le y\}) \\ &= \mu(\{x : F(x) \le G(y)\}) \\ &= \mu(\{x : x \le F^{-1}(G(y))\}) \\ &= F(F^{-1}(G(y))) = G(y),\end{aligned}$$

where the second and third lines apply the increasing bijections G and F^{-1} to both sides of the inequality. So $T_{\#}\mu$ and ν have the same distribution function, hence are equal.

Given optimality, the cost of T is W_p^p. By Lemma 1.1 with the substitution $x = F^{-1}(z)$, which carries Lebesgue measure on $[0, 1]$ to μ,

$$\int_{\mathbb{R}} |x - T(x)|^p \, d\mu(x) = \int_0^1 \left|F^{-1}(z) - G^{-1}(F(F^{-1}(z)))\right|^p dz = \int_0^1 \left|F^{-1}(z) - G^{-1}(z)\right|^p dz,$$

which is Equation 1.25. □

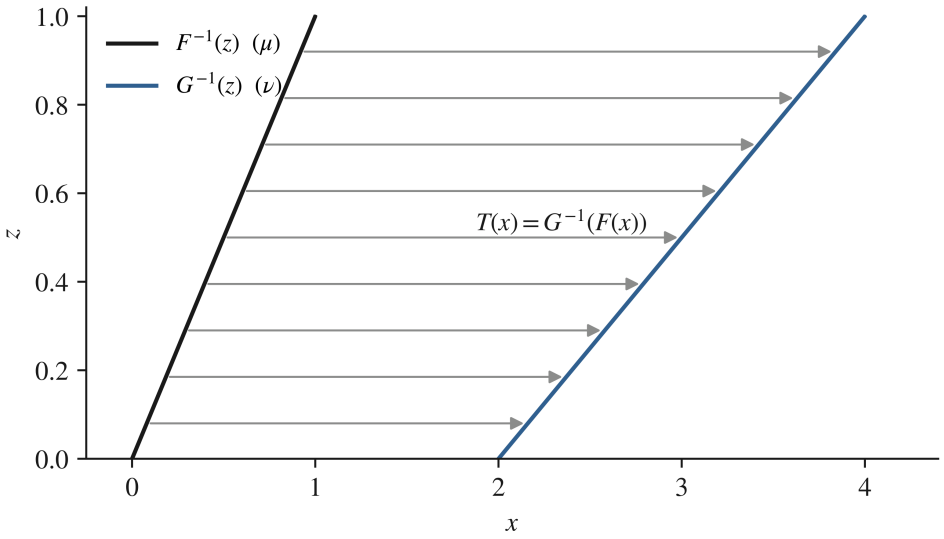


Figure 1.9.: Transport on the line. The map $T = G^{-1} \circ F$ carries the quantile of μ at level z to the quantile of ν at the same level, so the horizontal gap between the two inverse distribution functions is exactly the distance each unit of mass travels. Integrating the p-th power of that gap over z gives Equation 1.25.

In Figure 1.9 the map is read off horizontally: fix a level z, find the point that μ places at that quantile and the point that ν places at the same quantile, and the arrow between them is where that parcel of mass goes. The next example carries this through with numbers.

Example 1.2 (A transport computed by hand)**.** Let $\mu = \text{Uniform}[0,1]$ and $\nu = \text{Uniform}[2,4]$. Then $F(x) = x$ on $[0,1]$ and $G(y) = (y-2)/2$ on $[2,4]$, so $F^{-1}(z) = z$ and $G^{-1}(z) = 2 + 2z$, and the optimal map is

$$T(x) = G^{-1}(F(x)) = 2 + 2x.$$

The map stretches the unit interval to twice its length and shifts it. By Equation 1.25 the transport costs are

$$W_1 = \int_0^1 |z - (2+2z)|\, dz = \int_0^1 (2+z)\, dz = \frac{5}{2},$$

$$W_2 = \left(\int_0^1 (2+z)^2\, dz \right)^{1/2} = \left(\frac{(2+z)^3}{3} \Big|_0^1 \right)^{1/2} = \left(\frac{27-8}{3} \right)^{1/2} = \sqrt{\frac{19}{3}} \approx 2.5166.$$

Two sanity checks. The value $W_1 = 5/2$ equals the distance between the two means, $3 - 1/2$, as it must here because $T(x) - x = 2 + x$ never changes sign, so no mass moves against the flow. And $W_1 \le W_2$, as Jensen's inequality requires.

The computation is short enough to check numerically, which is worth doing once to see that the coupling really is optimal and not merely feasible:

```
import numpy as np
from scipy.stats import wasserstein_distance

rng = np.random.default_rng(0)
x = rng.uniform(0.0, 1.0, 2_000_000)      # mu  = Uniform[0, 1]
y = rng.uniform(2.0, 4.0, 2_000_000)      # nu  = Uniform[2, 4]

print(wasserstein_distance(x, y))         # ~2.5000   matches W_1 = 5/2
print(np.mean(np.abs(x - (2 + 2 * x))))   # ~2.5000   cost of T(x) = 2 + 2x
print(np.sqrt(np.mean((x - (2 + 2 * x)) ** 2)))   # ~2.5166  matches W_2
```

A second instance, with a map that is not affine, shows the two readings of Theorem 1.4 side by side.

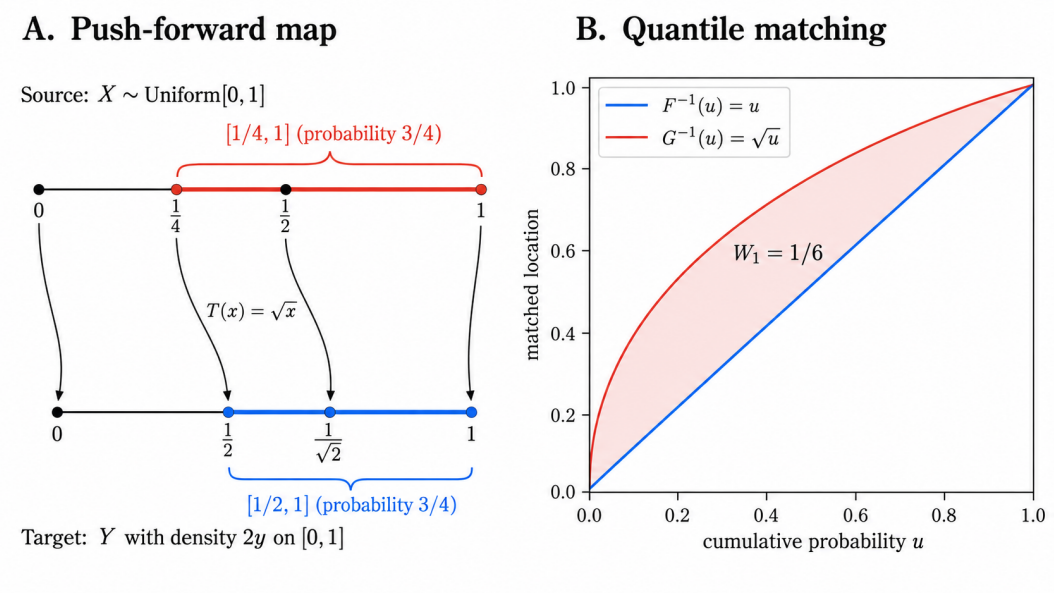


Figure 1.10.: One transport problem, seen twice. **A.** The push-forward of Equation 1.4 made concrete: $T(x) = \sqrt{x}$ carries $X \sim \text{Uniform}[0,1]$ to the law with density $2y$ on $[0,1]$, and the target event $[\frac{1}{2}, 1]$ receives exactly the mass its preimage $[\frac{1}{4}, 1]$ had, namely $\frac{3}{4}$. This T is the monotone map $G^{-1} \circ F$ of Theorem 1.4, since $F(x) = x$ and $G(y) = y^2$. **B.** The same pair through their quantile functions. Equation 1.25 integrates the horizontal gap between F^{-1} and G^{-1}, so W_1 is the shaded area between the two curves, here $\int_0^1 (\sqrt{u} - u)\, du = \frac{2}{3} - \frac{1}{2} = \frac{1}{6}$.

1.11. Kantorovich's relaxation

Kantorovich's reformulation replaces the map by a coupling, and so allows mass at a single point to be split between several destinations. It is the problem we already wrote down as Equation 1.22, now stated for a general cost.

Definition 1.13 (Kantorovich's problem)**.** Given $\mu \in \mathcal{P}(\mathcal{X})$, $\nu \in \mathcal{P}(\mathcal{Y})$ and a cost c,

$$\mathbb{K}(\pi) := \int_{\mathcal{X}\times\mathcal{Y}} c(x,y)\, d\pi(x,y), \qquad \min_{\pi\in\Pi(\mu,\nu)} \mathbb{K}(\pi). \tag{1.26}$$

Three things improve at once. The objective is linear in π, the constraint set $\Pi(\mu,\nu)$ is convex — a convex combination of couplings is a coupling, since Equation 1.21 is linear in π — and on finite spaces the whole thing is a linear program, the observation for which Kantorovich shared the 1975 Nobel Memorial Prize in Economics. Section 1.15.6 writes that program out, and reads off it both why the dual of Section 1.12 is the practical formulation and when the relaxation costs nothing.

Proposition 1.8 (The relaxation is genuine)**.** *Every transport map induces a coupling of the same cost, so*

$$\min_{\pi\in\Pi(\mu,\nu)} \mathbb{K}(\pi) \;\le\; \inf_{T:T_{\#}\mu=\nu} \mathbb{M}(T). \tag{1.27}$$

Proof. Let T satisfy $T_{\#}\mu = \nu$ and put $\pi_T = (\mathrm{id}, T)_{\#}\mu$, the law of $(X, T(X))$ for $X \sim \mu$. Its first marginal is μ because the first coordinate of (id, T) is the identity, and its second marginal is $T_{\#}\mu = \nu$; so $\pi_T \in \Pi(\mu,\nu)$. By Lemma 1.1 its cost is

$$\int c(x,y)\, d\pi_T(x,y) = \int c(x,T(x))\, d\mu(x) = \mathbb{M}(T).$$

Hence every value attained by $\mathbb{M}$ is attained by $\mathbb{K}$, and the infimum over the larger set is no larger. □

The inequality in Equation 1.27 can be strict, and Proposition 1.7 is the extreme case: the right-hand side is an infimum over the empty set, hence $+\infty$, while the left-hand side is finite. Indeed with $\mu = \delta_0$, $\nu = \frac{1}{2}(\delta_{-1} + \delta_1)$ and $c(x,y) = |x-y|$, the only coupling is $\pi = \frac{1}{2}\delta_{(0,-1)} + \frac{1}{2}\delta_{(0,1)}$ — it splits the atom, which no map can do — and its cost is 1. Figure 1.11 draws the two formulations on this example.

Unlike Monge's problem, Kantorovich's always has a solution.

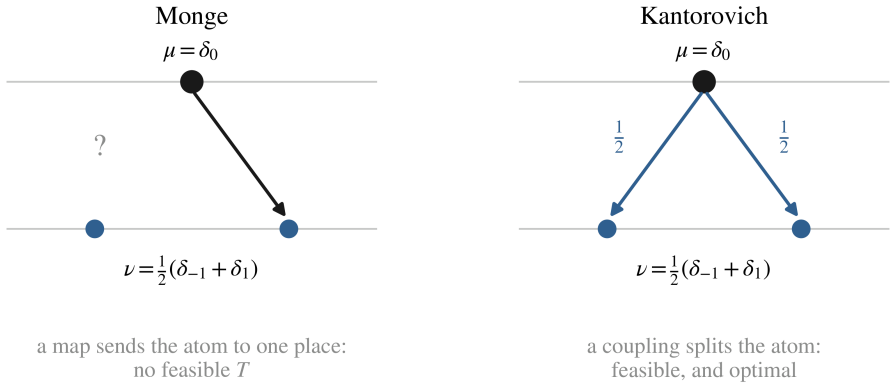


Figure 1.11.: Monge versus Kantorovich. On the left a transport map must send all the mass at a source point to a single destination, which is impossible when the source is one atom and the target is two. On the right a coupling may split that atom, and the resulting transport plan is feasible and optimal.

Theorem 1.5 (Existence of an optimal coupling)**.** *Let $\mathcal{X}, \mathcal{Y}$ be Polish spaces, $\mu \in \mathcal{P}(\mathcal{X})$, $\nu \in \mathcal{P}(\mathcal{Y})$, and let $c : \mathcal{X} \times \mathcal{Y} \to [0, \infty]$ be lower semicontinuous. Then the minimum in Equation 1.26 is attained.*

Proof. The set $\Pi(\mu, \nu)$ is non-empty, since $\mu \otimes \nu$ lies in it.

It is tight. Let $\varepsilon > 0$. By Ulam's theorem, Borel probability measures on a Polish space are tight, so there are compact sets $K_1 \subseteq \mathcal{X}$ and $K_2 \subseteq \mathcal{Y}$ with $\mu(K_1^c) \leq \varepsilon/2$ and $\nu(K_2^c) \leq \varepsilon/2$. For any $\pi \in \Pi(\mu, \nu)$, the complement of the compact set $K_1 \times K_2$ is contained in $(K_1^c \times \mathcal{Y}) \cup (\mathcal{X} \times K_2^c)$, so by Equation 1.21

$$\pi\big((K_1 \times K_2)^c\big) \leq \mu(K_1^c) + \nu(K_2^c) \leq \varepsilon.$$

The bound is uniform over $\Pi(\mu, \nu)$, so the family is tight and, by Prokhorov's theorem, relatively compact for the weak topology.

It is closed. If $\pi_n \to \pi$ weakly with $\pi_n \in \Pi(\mu, \nu)$, then for bounded continuous h on $\mathcal{X}$, the function $(x, y) \mapsto h(x)$ is bounded and continuous on the product, so $\int h(x) d\pi = \lim \int h(x) d\pi_n = \int h \, d\mu$. Hence the first marginal of π is μ, and symmetrically the second is ν. So $\Pi(\mu, \nu)$ is weakly closed, and with the previous paragraph, weakly compact.

The objective is lower semicontinuous. A non-negative lower semicontinuous c is the pointwise increasing limit of bounded continuous functions c_k. For each k, $\pi \mapsto \int c_k \, d\pi$ is weakly continuous, and a supremum of continuous functions is lower semicontinuous, so $\pi \mapsto \int c \, d\pi = \sup_k \int c_k \, d\pi$ is lower semicontinuous.

A lower semicontinuous function on a non-empty compact set attains its infimum, which is the claim. □

1.12. Kantorovich duality

Equation 1.26 is a linear program over an infinite-dimensional convex set, so it has a dual. The dual turns out to be the formulation that generative models actually implement, because it needs only the two marginals and never the joint measure.

Theorem 1.6 (Kantorovich duality)**.** *Let $\mu \in \mathcal{P}(\mathcal{X})$, $\nu \in \mathcal{P}(\mathcal{Y})$ with $\mathcal{X}, \mathcal{Y}$ Polish, and let $c : \mathcal{X} \times \mathcal{Y} \to [0, +\infty]$ be lower semicontinuous. Define*

$$\mathbb{J}(\varphi, \psi) = \int_{\mathcal{X}} \varphi \, d\mu + \int_{\mathcal{Y}} \psi \, d\nu, \qquad \Phi_c = \{(\varphi, \psi) \in L^1(\mu) \times L^1(\nu) : \varphi(x) + \psi(y) \le c(x, y)\}, \tag{1.28}$$

the inequality holding for μ-almost every x and ν-almost every y. Then

$$\min_{\pi \in \Pi(\mu,\nu)} \mathbb{K}(\pi) = \sup_{(\varphi,\psi) \in \Phi_c} \mathbb{J}(\varphi, \psi). \tag{1.29}$$

Proof. Weak duality, in general. Let $\pi \in \Pi(\mu, \nu)$ and $(\varphi, \psi) \in \Phi_c$. Because the marginals of π are μ and ν, integrating a function of x alone against π is the same as integrating it against μ, and likewise for y and ν. Hence

$$\mathbb{J}(\varphi, \psi) = \int \varphi(x) \, d\pi + \int \psi(y) \, d\pi = \int (\varphi(x) + \psi(y)) d\pi \le \int c(x, y) \, d\pi = \mathbb{K}(\pi),$$

the inequality being the constraint defining Φ_c integrated against the non-negative measure π. Taking the supremum on the left and the infimum on the right gives $\sup \mathbb{J} \le \min \mathbb{K}$.

Strong duality, on finite spaces. Let $\mathcal{X} = \{x_1, \dots, x_m\}$ and $\mathcal{Y} = \{y_1, \dots, y_n\}$, write μ_i, ν_j, c_{ij} and π_{ij}. The problem is the linear program

$$\min_{\pi \ge 0} \sum_{i,j} c_{ij} \pi_{ij} \quad \text{subject to} \quad \sum_j \pi_{ij} = \mu_i, \quad \sum_i \pi_{ij} = \nu_j.$$

Attach multipliers φ_i and ψ_j to the two families of constraints. The Lagrangian is

$$\begin{aligned} L(\pi, \varphi, \psi) &= \sum_{i,j} c_{ij} \pi_{ij} + \sum_i \varphi_i \Big(\mu_i - \sum_j \pi_{ij} \Big) + \sum_j \psi_j \Big(\nu_j - \sum_i \pi_{ij} \Big) \\ &= \sum_{i,j} \pi_{ij} (c_{ij} - \varphi_i - \psi_j) + \sum_i \varphi_i \mu_i + \sum_j \psi_j \nu_j. \end{aligned}$$

Minimising over $\pi \geq 0$ sends the first sum to $-\infty$ unless every coefficient is non-negative, and gives 0 when they all are. So the dual function is finite exactly on $\{\varphi_i + \psi_j \leq c_{ij}\}$, where it equals $\mathbb{J}(\varphi, \psi)$, and the dual problem is $\sup_{\Phi_c} \mathbb{J}$. The primal is feasible, since $\mu \otimes \nu$ is a coupling, and bounded below by 0, so linear programming duality gives equality of the two optimal values. The passage from finite spaces to Polish spaces is by approximation and is given in Villani, Theorem 1.3. □

The dual constraint couples φ and ψ, but only in one direction: for a fixed φ there is a best possible ψ, and it can be written down.

Definition 1.14 (c-transform and c-concavity)**.** For $\varphi : \mathcal{X} \to \mathbb{R}$ and $\psi : \mathcal{Y} \to \mathbb{R}$, the c-transforms are

$$\varphi^c(y) := \inf_x \{c(x, y) - \varphi(x)\}, \qquad \psi^c(x) := \inf_y \{c(x, y) - \psi(y)\}. \tag{1.30}$$

A function φ is called c-concave if $\varphi = \psi^c$ for some ψ. Optimal φ and ψ are called Kantorovich potentials.

Proposition 1.9 (The dual is a supremum over one potential)**.** *The dual problem in Equation 1.29 can be written*

$$\sup_{(\varphi,\psi)\in\Phi_c} \mathbb{J}(\varphi, \psi) = \sup_{\varphi} \left\{ \int_{\mathcal{X}} \varphi \, d\mu + \int_{\mathcal{Y}} \varphi^c \, d\nu \right\} = \sup_{\psi} \left\{ \int_{\mathcal{X}} \psi^c \, d\mu + \int_{\mathcal{Y}} \psi \, d\nu \right\}, \tag{1.31}$$

and the supremum may be restricted to c-concave φ.

Proof. Fix φ. The constraint $(\varphi, \psi) \in \Phi_c$ says $\psi(y) \leq c(x, y) - \varphi(x)$ for every x, which is exactly $\psi(y) \leq \varphi^c(y)$ by Equation 1.30. Since $\mathbb{J}$ is increasing in ψ and $\nu \geq 0$, the largest admissible choice $\psi = \varphi^c$ is optimal. This gives the first equality in Equation 1.31, and the second follows by the symmetric argument with the roles of the two potentials exchanged.

For the restriction, apply the transform twice. For every x and y, $\varphi^c(y) \leq c(x, y) - \varphi(x)$, so $c(x, y) - \varphi^c(y) \geq \varphi(x)$; taking the infimum over y gives

$$\varphi^{cc}(x) = \inf_y \{c(x, y) - \varphi^c(y)\} \geq \varphi(x).$$

So replacing φ by φ^{cc} never decreases $\int \varphi \, d\mu$. Applying the same inequality to φ^c gives $\varphi^{ccc} \geq \varphi^c$, while $\varphi^{cc} \geq \varphi$ and the order-reversing nature of the transform give $\varphi^{ccc} \leq \varphi^c$; hence $\varphi^{ccc} = \varphi^c$ and the second term is unchanged. So $(\varphi^{cc}, \varphi^c)$ is admissible, has objective value at least that of (φ, φ^c), and φ^{cc} is c-concave by construction. □

1.12.1. The Kantorovich–Rubinstein form of the Wasserstein-1 distance

Everything above holds for a general cost. For the particular cost $c(x, y) = \|x - y\|$ the c-transform collapses, and the dual takes the form that W-GAN implements.

Lemma 1.3 (The c-transform of a 1-Lipschitz function)**.** *Let $c(x, y) = \|x - y\|$ and let $\varphi \in \mathrm{Lip}_1(\mathcal{X})$. Then $\varphi^c = -\varphi$. Moreover a function is c-concave for this cost if and only if it is 1-Lipschitz, so the two classes coincide.*

Proof. From Equation 1.30, choosing the particular point $x = y$ in the infimum,

$$\varphi^c(y) = \inf_x \{\|x - y\| - \varphi(x)\} \le \|y - y\| - \varphi(y) = -\varphi(y).$$

In the other direction, the 1-Lipschitz property gives $\varphi(x) \le \varphi(y) + \|x - y\|$ for every x, hence $\|x - y\| - \varphi(x) \ge -\varphi(y)$ for every x, and taking the infimum over x,

$$\varphi^c(y) \ge -\varphi(y).$$

The two inequalities give $\varphi^c = -\varphi$.

For the second claim, a c-concave function is by definition $\psi^c(x) = \inf_y\{\|x - y\| - \psi(y)\}$, an infimum of functions of x each of which is 1-Lipschitz; an infimum of a family of 1-Lipschitz functions is 1-Lipschitz wherever it is finite. Conversely a 1-Lipschitz φ is c-concave: taking $\psi = -\varphi$,

$$\psi^c(x) = \inf_y \{\|x - y\| + \varphi(y)\} = \varphi(x),$$

because $\varphi(y) \ge \varphi(x) - \|x - y\|$ makes every term at least $\varphi(x)$, with equality at $y = x$. So the two classes coincide. □

Theorem 1.7 (Kantorovich–Rubinstein)**.** *For $c(x, y) = \|x - y\|$,*

$$W_1(\mu, \nu) = \min_{\pi \in \Pi(\mu,\nu)} \int \|x - y\| \, d\pi(x, y) = \sup_{\varphi \in \mathrm{Lip}_1(\mathcal{X})} \left\{ \int_{\mathcal{X}} \varphi \, d\mu - \int_{\mathcal{X}} \varphi \, d\nu \right\}. \tag{1.32}$$

Proof. The first equality is Definition 1.10 with $p = 1$ and $d(x, y) = \|x - y\|$. For the second, start from Equation 1.31, whose supremum may be taken over c-concave φ only. By Lemma 1.3 the c-concave functions for this cost are exactly the 1-Lipschitz ones, and on that class $\varphi^c = -\varphi$. Substituting,

$$\sup_{\varphi}\left\{\int \varphi\, d\mu + \int \varphi^{c}\, d\nu\right\} = \sup_{\varphi\in \mathrm{Lip}_1}\left\{\int \varphi\, d\mu - \int \varphi\, d\nu\right\},$$

and Theorem 1.6 equates this with the primal minimum. □

Compare Equation 1.32 with Definition 1.8. The right-hand side is exactly the integral probability metric $d_{\mathcal{F}}$ with $\mathcal{F} = \mathrm{Lip}_1(\mathcal{X})$ — the case promised in Section 1.8 and left unproved there. One step is needed to match the two expressions, since Definition 1.8 carries an absolute value and Equation 1.32 does not, and it is the symmetry assumed in Section 1.8: Lip_1 is closed under $\varphi \mapsto -\varphi$, so the supremum of the signed difference already equals the supremum of its absolute value.

The two families of statistical distance introduced in this chapter, one built from density ratios and one built from transport cost, meet at this single identity: W_1 is simultaneously the cheapest way to move mass and the most a 1-Lipschitz test function can tell the two distributions apart.

The practical consequence is the one the lecture stresses. The primal in Equation 1.32 requires optimising over joint measures on $\mathcal{X} \times \mathcal{X}$, which is hopeless in high dimension. The dual requires only a single function φ and two expectations, each estimated from a sample of the corresponding marginal. Parameterise φ by a network constrained to be 1-Lipschitz, maximise the bracket in Equation 1.32, and the result estimates W_1; that network is the W-GAN critic, and the Lipschitz constraint is why it is clipped or gradient-penalised rather than left free.

1.13. Brenier's theorem and the Gaussian case

Kantorovich's relaxation buys existence at the price of abandoning maps. For the quadratic cost the map comes back, and with a great deal of extra structure.

Theorem 1.8 (Brenier)**.** *Let $c(x,y) = \frac{1}{2}\|x-y\|^2$ on $\mathbb{R}^d$, and let μ, ν have finite second moments with μ absolutely continuous with respect to Lebesgue measure. Then the optimal coupling of Equation 1.26 is unique and is induced by a map: there is a convex function u such that $T = \nabla u$ satisfies $T_{\#}\mu = \nu$ and $\pi = (\mathrm{id}, T)_{\#}\mu$ is optimal. The map T is unique μ-almost everywhere.*

Proof. Expand the cost as $c(x,y) = \frac{1}{2}\|x\|^2 - \langle x, y\rangle + \frac{1}{2}\|y\|^2$ and change variables in the dual, setting

$$u(x) = \tfrac{1}{2}\|x\|^2 - \varphi(x), \qquad v(y) = \tfrac{1}{2}\|y\|^2 - \psi(y).$$

The dual constraint $\varphi(x) + \psi(y) \le c(x,y)$ becomes, after substituting and cancelling the two quadratic terms,

$$\langle x, y\rangle \le u(x) + v(y), \tag{1.33}$$

which is precisely the Fenchel–Young inequality for the pair u, v. In these variables the c-transform of Definition 1.14 is the Legendre transform, the relation $\psi = \varphi^c$ reading $v = u^*$; this is worked out in Section 1.15.9. By Proposition 1.9 the supremum may be restricted to c-concave φ, which says here that u may be taken convex and lower semicontinuous with $u = v^*$.

Let π be an optimal coupling and (φ, ψ) optimal potentials, which exist for this cost because $\frac{1}{2}\|x-y\|^2 \le \|x\|^2 + \|y\|^2$ and both measures have finite second moments; Section 1.15.7 states the hypothesis in general. Weak duality was an integrated inequality, and equality of the two optimal values forces the non-negative integrand $c - \varphi - \psi$ to vanish π-almost everywhere. So Equation 1.33 holds with equality for π-almost every (x, y):

$$\langle x, y\rangle = u(x) + v(y).$$

The case of equality in the Fenchel–Young inequality is exactly $y \in \partial u(x)$.

Next we check that u is finite and differentiable where the argument needs it. This deserves a step rather than an appeal: $u = v^*$ is a supremum of affine functions and may perfectly well be $+\infty$ off its effective domain. The equality above holds on a set of full π-measure, whose projection to the first coordinate therefore has full μ-measure, and on it $u(x) = \langle x, y\rangle - v(y)$ is finite. A convex function finite at each point of a set is finite on that set's convex hull, so $\operatorname{dom} u$ contains the convex hull K of a set of full μ-measure. Since $\mu \ll$ Lebesgue cannot charge a hyperplane, K has non-empty interior; and the boundary of a convex set in $\mathbb{R}^d$ is Lebesgue-null, so $\mu(\partial K) = 0$ and μ-almost every x lies in $\operatorname{int} K \subseteq \operatorname{int}\operatorname{dom} u$. On the interior of its domain a finite convex function is locally Lipschitz, hence differentiable outside a set of Lebesgue measure zero by Rademacher's theorem; that exceptional set is μ-null too. So for μ-almost every x the subdifferential is a single point, $\partial u(x) = \{\nabla u(x)\}$. Therefore $y = \nabla u(x)$ for π-almost every (x, y), which says π is concentrated on the graph of $T = \nabla u$ and hence $\pi = (\mathrm{id}, T)_{\#}\mu$. Its second marginal is $T_{\#}\mu$ and must equal ν.

Uniqueness. Everything above applies to *any* optimal coupling, so each one is concentrated on the graph of the gradient of some convex potential. Let π and π' be optimal, carried by the graphs of T and T'. The objective Equation 1.26 is linear and $\Pi(\mu,\nu)$ convex, so $\frac{1}{2}(\pi + \pi')$ is optimal as well, hence also carried by a graph; a measure carried by a graph assigns to μ-almost every x a single y, which forces $T = T'$ μ-almost everywhere. That gives uniqueness of both the map and the plan. Note what is *not* being claimed: the potential u itself is pinned down only up to an additive constant on each connected

component of the interior of its domain, so on a disconnected support one may shift the components independently without changing anything. It is ∇u, not u, that is unique. □

The factor $\frac{1}{2}$ in the cost is only a normalisation. Multiplying c by a positive constant rescales the optimal value but leaves the minimising coupling unchanged, so the map of Theorem 1.8 is equally optimal for $c(x, y) = \|x - y\|^2$, which is the cost Definition 1.10 uses at $p = 2$; the distance computed below is for that unhalved cost.

The theorem says something striking: the optimal transport map for the quadratic cost is never arbitrary. It is always the gradient of a convex potential, so it is a curl-free, monotone rearrangement of space. The lecture's Gaussian formula is the simplest instance where the potential can be written in closed form.

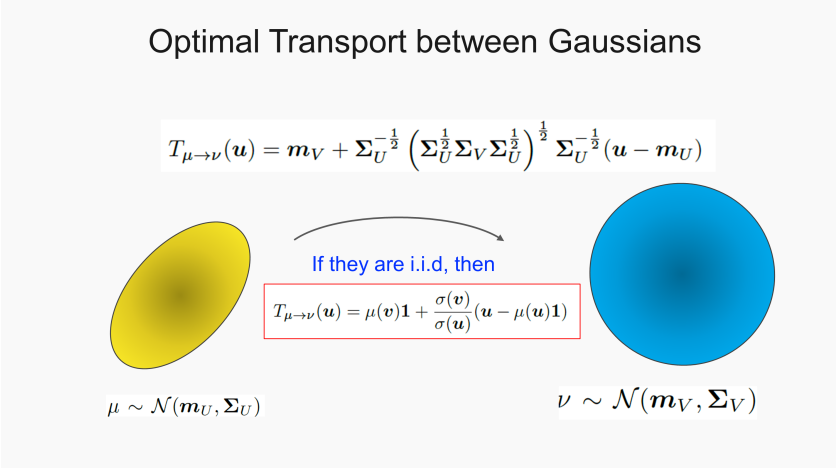


Figure 1.12.: Transport between two Gaussians. The map of Proposition 1.10 carries μ on the left to ν on the right by matching means and reshaping the covariance. The boxed special case is what happens when both covariances are isotropic: it is Equation 1.36, the layer that normalises a feature map and rescales it to another's statistics. From the course slides (Ye 2026a, 40).

Figure 1.12 is worth keeping in view through the next two sections, because the step from the general map to the boxed one is the whole of the connection to adaptive instance normalisation: the matrix square roots collapse to a ratio of scalars the moment the covariances commute.

Proposition 1.10 (Optimal transport between Gaussians)**.** *Let* $\mu = \mathcal{N}(m_U, \Sigma_U)$ *and* $\nu = \mathcal{N}(m_V, \Sigma_V)$ *with* Σ_U *positive definite. Put*

$$A = \Sigma_U^{-1/2} (\Sigma_U^{1/2} \Sigma_V \Sigma_U^{1/2})^{1/2} \Sigma_U^{-1/2}, \qquad T_{\mu\to\nu}(u) = m_V + A\,(u - m_U). \tag{1.34}$$

Then A is symmetric positive semi-definite, $T_{\mu\to\nu}$ is the gradient of a convex function, $T_{\mu\to\nu\#}\mu = \nu$, and

$$W_2^2(\mu,\nu) = \|m_U - m_V\|^2 + \operatorname{tr}\Big(\Sigma_U + \Sigma_V - 2(\Sigma_U^{1/2}\Sigma_V\Sigma_U^{1/2})^{1/2}\Big). \tag{1.35}$$

Proof. Write $S = \Sigma_U^{1/2}$, symmetric positive definite, and $M = (S\Sigma_V S)^{1/2}$, symmetric positive semi-definite, so that $A = S^{-1}MS^{-1}$.

Symmetry and positivity. $A^\top = (S^{-1})^\top M^\top (S^{-1})^\top = S^{-1}MS^{-1} = A$, and for any z, $z^\top Az = (S^{-1}z)^\top M(S^{-1}z) \geq 0$.

Gradient of a convex function. Set $\phi(u) = \frac{1}{2}u^\top Au + (m_V - Am_U)^\top u$. Then $\nabla\phi(u) = Au + m_V - Am_U = T_{\mu\to\nu}(u)$, and ϕ is convex because A is positive semi-definite. By Theorem 1.8 this identifies $T_{\mu\to\nu}$ as the optimal map, provided it pushes μ to ν.

Push-forward. If $u \sim \mathcal{N}(m_U, \Sigma_U)$ then $T_{\mu\to\nu}(u)$ is Gaussian with mean m_V and covariance $A\Sigma_U A$. Compute, using $\Sigma_U = S^2$ and $M^2 = S\Sigma_V S$:

$$A\Sigma_U A = S^{-1}MS^{-1}\cdot S^2 \cdot S^{-1}MS^{-1} = S^{-1}M^2S^{-1} = S^{-1}S\Sigma_V SS^{-1} = \Sigma_V.$$

So the push-forward is $\mathcal{N}(m_V, \Sigma_V) = \nu$.

The distance. By Proposition 1.8 and Theorem 1.8 the optimal cost is attained at this map, so with $u \sim \mu$ and $w = u - m_U$,

$$\begin{aligned} W_2^2(\mu,\nu) &= \mathbb{E}\big\|u - T_{\mu\to\nu}(u)\big\|^2 = \mathbb{E}\big\|(m_U - m_V) + (I - A)w\big\|^2 \\ &= \|m_U - m_V\|^2 + \operatorname{tr}\big((I-A)\Sigma_U(I-A)\big), \end{aligned}$$

the cross term vanishing because $\mathbb{E}w = 0$. Expanding the trace,

$$\operatorname{tr}(\Sigma_U - 2A\Sigma_U + A\Sigma_U A) = \operatorname{tr}\Sigma_U - 2\operatorname{tr}(A\Sigma_U) + \operatorname{tr}\Sigma_V,$$

using $A\Sigma_U A = \Sigma_V$ from above and the symmetry of A and Σ_U. Finally, by cyclicity of the trace, $\operatorname{tr}(A\Sigma_U) = \operatorname{tr}(S^{-1}MS^{-1}S^2) = \operatorname{tr}(S^{-1}MS) = \operatorname{tr}(M)$, which is $\operatorname{tr}((\Sigma_U^{1/2}\Sigma_V\Sigma_U^{1/2})^{1/2})$. Substituting gives Equation 1.35. □

Example 1.3 (The isotropic case is adaptive instance normalisation)**.** Suppose the two Gaussians are isotropic, $\Sigma_U = \sigma(u)^2 I$ and $\Sigma_V = \sigma(v)^2 I$. Then $S = \sigma(u)I$, $M = (\sigma(u)^2\sigma(v)^2 I)^{1/2} = \sigma(u)\sigma(v)I$, and

$$A = \frac{1}{\sigma(u)}\cdot\sigma(u)\sigma(v)\cdot\frac{1}{\sigma(u)}I = \frac{\sigma(v)}{\sigma(u)}I,$$

so Equation 1.34 becomes

$$T_{\mu\to\nu}(u) = \mu(v)\mathbf{1} + \frac{\sigma(v)}{\sigma(u)}(u - \mu(u)\mathbf{1}), \tag{1.36}$$

writing $\mu(\cdot)$ and $\sigma(\cdot)$ for the scalar mean and standard deviation. Equation 1.36 is exactly the adaptive instance normalisation layer used in style transfer: standardise the content features and rescale them by the statistics of the style features. So AdaIN is not a heuristic — it is the optimal transport map between two isotropic Gaussians, and it will reappear under that name in the lecture on StyleGAN.

The Gaussian formulas are easy to get wrong by a transpose, so they are worth checking once numerically:

```
import numpy as np
from scipy.linalg import sqrtm

rng = np.random.default_rng(0)
d = 4
def rand_cov():
    B = rng.normal(size=(d, d))
    return B @ B.T + d * np.eye(d)

mU, mV = rng.normal(size=d), rng.normal(size=d)
SU, SV = rand_cov(), rand_cov()

S  = np.real(sqrtm(SU))
Si = np.linalg.inv(S)
M  = np.real(sqrtm(S @ SV @ S))
A  = Si @ M @ Si                                  # equation (gaussian-map)

print(np.allclose(A, A.T))                        # True: A is symmetric
print(np.linalg.eigvalsh(A).min() > 0)            # True: A is positive definite
print(np.allclose(A @ SU @ A, SV))                # True: pushes mu to nu

w2 = np.sum((mU - mV) ** 2) + np.trace(SU + SV - 2 * M)
u  = rng.multivariate_normal(mU, SU, 400_000)
t  = mV + (u - mU) @ A.T                          # A symmetric, so A.T == A
print(w2, np.mean(np.sum((u - t) ** 2, axis=1)))  # the two agree
```

1.14. Back to generative models

We can now close the loop opened in Section 1.3. A generator defines $\mu_\theta = G_{\theta\#}\nu$ by Equation 1.1, and training means minimising some discrepancy $D(\mu_\theta, p_{\text{data}})$ over θ. This chapter has supplied two families of candidates for D, and the choice between them is an optimisation decision rather than a matter of taste.

Take D to be an f-divergence and one obtains the adversarial models. The original GAN objective, at its optimal discriminator, reduces to the Jensen–Shannon divergence between μ_θ and p_{data} up to an additive constant; f-GAN generalises this by replacing the generator f and deriving the corresponding variational bound. Theorem 1.2 is then the warning label on the whole construction: as soon as the discriminator becomes good enough to separate the two supports, the objective is pinned at $f(0) + f'(\infty)$ and its gradient in θ vanishes.

Take D to be an integral probability metric and one obtains the alternatives. W-GAN takes $\mathcal{F} = \text{Lip}_1$ and optimises the dual in Equation 1.32, which is why its critic is Lipschitz-constrained and why its loss remains informative when the supports are disjoint — by Example 1.1 it still reports $|\theta|$. The general statement behind that example — that the loss is continuous in θ, where the Jensen–Shannon one is not — is Section 1.15.10. McGAN and MMD-GAN take $\mathcal{F}$ to be a moment class or an RKHS ball and use Equation 1.20, which is estimable directly from two samples and needs no inner maximisation at all.

Discrepancy	Metric	Needs $P \ll Q$	Value when $P \perp Q$	Used by
D_{KL}	no	yes	$+\infty$	maximum likelihood
Reverse D_{KL}	no	yes	$+\infty$	variational inference
D_{JS}	no	no	$\log 2$	GAN, f-GAN
Total variation	yes	no	1	analysis, rarely trained
MMD	yes	no	varies with the kernel	MMD-GAN, McGAN
W_1	yes	no	varies with the distance	W-GAN

The last two rows are the point of the table. Only for the integral probability metrics does the entry in the fourth column depend on the configuration of the two distributions rather than on the discrepancy alone, and that dependence is precisely the gradient signal a generator needs. Figure 1.13 collects the whole development onto one page.

Three threads continue directly into the next lectures. The reduction of the GAN objective to the Jensen–Shannon divergence, and the f-GAN variational bound, are the subject of

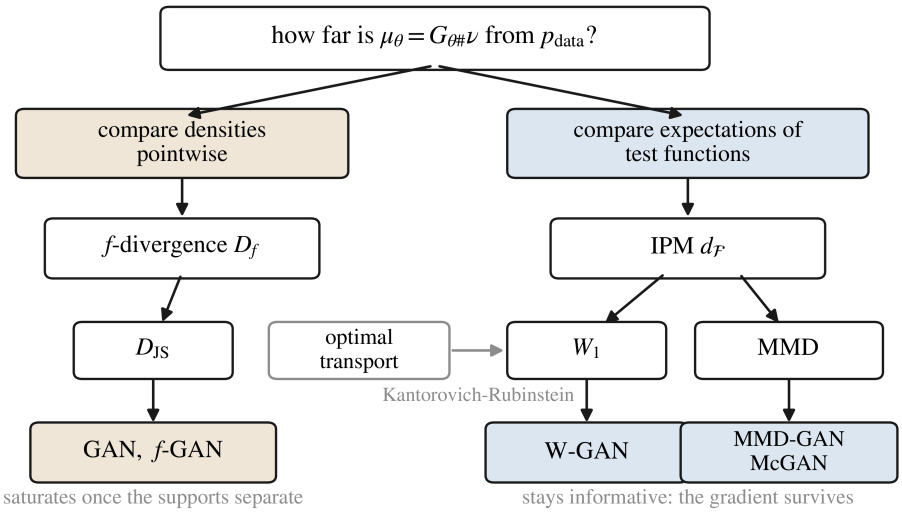


Figure 1.13.: The two routes from a statistical distance to a generative model. Reading down the left branch gives the adversarial family and its saturation problem; reading down the right gives the transport and moment-matching families. Kantorovich–Rubinstein duality, Equation 1.32, is the bridge between the two branches.

Lecture 02, as are W-GAN, McGAN and the geometric view. Equation 1.36, the optimal map between isotropic Gaussians, returns as the adaptive instance normalisation layer of StyleGAN in Lecture 03. And the idea that a generative model is a transport of one measure onto another — rather than a likelihood to be maximised — is the thread that runs through the diffusion, flow matching and Schrodinger bridge lectures in the rest of the course.

1.15. Advanced topics

The chapter above follows the lecture. This section collects material that the lecture did not cover but that the development above leans on or leaves open: a divergence the tables list without evaluating, the two estimators that make the maximum mean discrepancy usable, the linear-programming form of the Kantorovich problem, the hypotheses under which duality survives the passage to Polish spaces, and the topological facts that turn Example 1.1 from an example into a theorem about training. Each subsection is self-contained and none is needed to follow the main line.

1.15.1. Where the Pearson chi-squared divergence is finite

The generator table of Section 1.6 lists $f(u) = (u-1)^2$, and the saturation table of Section 1.7 gives its row, but neither is worked through in the chapter. Here $f(0) = 1$ and $f'(\infty) = \lim_{u\to\infty}(u-1)^2/u = +\infty$, so by Theorem 1.2 the Pearson χ^2 divergence is $+\infty$ on mutually singular pairs, joining the two Kullback–Leibler divergences rather than the bounded generators.

Away from that case,

$$D_{\chi^2}(P\|Q) = \int \left(\frac{dP}{dQ} - 1\right)^2 dQ = \int \left(\frac{dP}{dQ}\right)^2 dQ \;-\; 1, \tag{1.37}$$

so χ^2 is finite exactly when $P \ll Q$ **and** $dP/dQ \in L^2(Q)$. That is a strictly stronger demand than the absolute continuity D_{KL} needs, and it is why χ^2 plays no part in the generative models of this course: the situation of Section 1.7 defeats it even more thoroughly than the others.

1.15.2. How literal is the dual-norm reading?

Definition 1.8 was introduced as the norm of the signed measure $\mu - \nu$ measured over a unit ball. The reading is exact for two of the three classes of Section 1.8 and only approximate for the third, and the difference is worth pinning down because it is the reason Theorem 1.7 needs a normalisation.

If $\|\cdot\|_B$ is a genuine norm on a space B of test functions and $\mathcal{F} = \{h : \|h\|_B \le 1\}$, then

$$d_{\mathcal{F}}(\mu,\nu) = \|\mu - \nu\|_{B^*} \tag{1.38}$$

by the definition of the dual norm. The indicator class and the RKHS ball of Proposition 1.6 are of this kind. The class Lip_1 is not: the Lipschitz constant vanishes on constant functions, so it is a seminorm rather than a norm, and Lip_1 is unbounded in any function norm — it contains $\varphi + t$ for every real t. The dual reading is recovered by quotienting out the constants, which is legitimate here because $\mu - \nu$ annihilates them, both measures having total mass 1; equivalently, by normalising $\varphi(x_0) = 0$ at a base point, which is what one does in practice when implementing Equation 1.32.

The choice of $\mathcal{F}$ also fixes a topology on $\mathcal{P}(\mathcal{X})$, and the three are genuinely different:

$\mathcal{F}$	Metric	Topology induced
indicators $\mathbf{1}_A$	total variation	strictly finer than weak; the one Theorem 1.2 bites in
$\{h : \|h\|_\infty + \mathrm{Lip}(h) \le 1\}$	bounded-Lipschitz	weak convergence

$\mathcal{F}$	Metric	Topology induced
Lip_1	W_1	weak convergence, on a space of bounded diameter

The second row is the Dudley metric. The third is what Section 1.15.4 takes up in general, and the gap between the first row and the other two is the whole argument of Section 1.14.

1.15.3. Estimating the maximum mean discrepancy

Equation 1.20 writes MMD^2 entirely in terms of expectations of the kernel, which is what makes it usable: no density estimate and no inner maximisation. Given $x_1, \dots, x_m \sim \mu$ and $y_1, \dots, y_n \sim \nu$ independently, replacing each expectation by the corresponding sample average gives the V-statistic

$$\widehat{\mathrm{MMD}}^2_V = \frac{1}{m^2}\sum_{i,j} k(x_i, x_j) - \frac{2}{mn}\sum_{i,j} k(x_i, y_j) + \frac{1}{n^2}\sum_{i,j} k(y_i, y_j). \tag{1.39}$$

It is biased upwards, because the diagonal terms $k(x_i, x_i)$ estimate $\mathbb{E}_{x,x'} k(x, x')$ with $x' = x$ rather than with an independent copy. Deleting the diagonals gives the U-statistic

$$\widehat{\mathrm{MMD}}^2_U = \frac{1}{m(m-1)}\sum_{i\neq j} k(x_i, x_j) - \frac{2}{mn}\sum_{i,j} k(x_i, y_j) + \frac{1}{n(n-1)}\sum_{i\neq j} k(y_i, y_j), \tag{1.40}$$

which is unbiased for MMD^2 but is no longer a squared norm, so a single estimate can come out negative when the two distributions agree. Both need $O((m+n)^2)$ kernel evaluations. That quadratic cost is what keeps the maximum mean discrepancy to moderate batch sizes, and is the practical reason it is more often used as a two-sample *test* statistic than as a training loss at scale.

Whether the maximum mean discrepancy separates distributions at all is a property of the kernel and not of the estimator. Call k **characteristic** if the mean embedding $\mu \mapsto m_\mu$ is injective on $\mathcal{P}(\mathcal{X})$; by Proposition 1.5 this is exactly the condition under which $\mathrm{MMD}(\mu, \nu) = 0$ if and only if $\mu = \nu$, that is, under which Equation 1.19 is a metric and not merely a pseudo-metric. For a translation-invariant kernel on $\mathbb{R}^d$ the condition can be read off the spectrum. Bochner's theorem writes

$$k(x, y) = \int_{\mathbb{R}^d} e^{i\langle \omega,\, x - y\rangle}\, d\Lambda(\omega) \tag{1.41}$$

for a finite non-negative spectral measure Λ, and k is characteristic exactly when $\operatorname{supp}(\Lambda) = \mathbb{R}^d$: the embedding then sees every frequency of the characteristic function of $\mu - \nu$, so no non-zero difference can be invisible. The Gaussian kernel qualifies, its spectral measure being Gaussian and hence of full support. A band-limited kernel does not, and is blind to whatever the two distributions do above its cutoff.

1.15.4. The topology of the Wasserstein distance

Theorem 1.3 establishes that W_p is a metric. That alone is not the property Section 1.14 relies on; what matters is which notion of convergence it defines. On a Polish space,

$$W_p(\mu_n, \mu) \to 0 \iff \mu_n \to \mu \text{ weakly} \quad \text{and} \quad \int d(x, x_0)^p d\mu_n \to \int d(x, x_0)^p d\mu \tag{1.42}$$

for some — equivalently any — base point x_0. On a space of bounded diameter, which is the case for images confined to a fixed pixel range, the moment condition is automatic and W_p metrises weak convergence outright.

No f-divergence does this. Total variation and the Jensen–Shannon divergence induce a strictly finer topology, and Example 1.1 is the gap made concrete: $\mu_\theta \to \mu_0$ weakly as $\theta \to 0$, yet both stay pinned at their saturation values until the instant $\theta = 0$. That mismatch of topologies, rather than the arithmetic of any single example, is the precise sense in which the Wasserstein distance "still varies smoothly" where the others do not, and Section 1.15.10 turns it into a statement about training.

1.15.5. Two further ways Monge's problem fails

Proposition 1.7 rules out a transport map by an atom: a map cannot split mass. It is easy to read that as the only difficulty, and to conclude that a non-atomic μ makes Equation 1.24 well behaved. It does not. The feasible set $\{T : T_{\#}\mu = \nu\}$ is not closed in any topology for which $\mathbb{M}$ is continuous, so the infimum can fail to be attained even when feasible maps are plentiful.

Example 1.4 (An infimum with no minimiser)**.** Work in $\mathbb{R}^2$ with $c(x, y) = \|x - y\|$. Let μ be uniform on the segment $\{0\} \times [0, 1]$, and let $\nu = \frac{1}{2}\nu_- + \frac{1}{2}\nu_+$ with $\nu_\pm$ uniform on $\{\pm 1\} \times [0, 1]$. Then $\inf_T \mathbb{M}(T) = 1$, but $\mathbb{M}(T) > 1$ for every feasible T, so no optimal map exists.

Proof. Every feasible map costs more than 1. A point $x = (0, t)$ and a point $y = (\pm 1, s)$ of the target are at distance $\sqrt{1 + (t - s)^2} \geq 1$, with equality only when $s = t$. So $\mathbb{M}(T) \geq 1$ always, with equality only if T preserves height μ-almost everywhere. But a

height-preserving T pushes μ onto a set meeting each height at most once, whereas ν charges two points at every height; such a T is not feasible. Hence $\mathbb{M}(T) > 1$.

The infimum is 1. Cut $[0,1]$ into $2n$ strips of height $1/(2n)$ and let T_n send the odd strips to the left target and the even strips to the right, each by the affine map onto the corresponding strip of height $1/n$. Each half receives mass $n \cdot \frac{1}{2n} = \frac{1}{2}$ spread uniformly, so T_n is feasible, and it moves any point vertically by at most $1/n$, so $\mathbb{M}(T_n) \le \sqrt{1+n^{-2}} \to 1$.

Kantorovich attains it. The plan that splits each $(0,t)$ evenly between $(-1,t)$ and $(1,t)$ is a coupling of cost exactly 1. □

This example also sharpens Equation 1.27: there the inequality is strict with *both* sides finite, which Proposition 1.7, whose right-hand side is an infimum over the empty set, could not show.

Failure of uniqueness is a third phenomenon, independent of the other two. For Monge's original cost $c(x,y) = \|x-y\|$, take $\mu = \text{Uniform}[0,1]$ and $\nu = \text{Uniform}[2,3]$ on the line. Every target point lies to the right of every source point, so $|x-y| = y-x$ and every feasible map has the same cost $\mathbb{E}_\nu[y] - \mathbb{E}_\mu[x] = 2$. All of them are optimal, including the monotone map of Theorem 1.4 and including maps that reverse order. Uniqueness, when it holds, comes from strict convexity of the cost — it is the quadratic cost of Theorem 1.8 that forces a single answer, not transport as such.

1.15.6. The Kantorovich problem as a linear program

On finite spaces Equation 1.26 is a linear program, and it is worth displaying rather than leaving inside the proof of Theorem 1.6: it is what a solver actually receives, and its shape is the reason the dual of Section 1.12, and not the primal, is what generative models implement. Take $\mathcal{X} = \{x_1, \dots, x_m\}$ and $\mathcal{Y} = \{y_1, \dots, y_n\}$, collect the marginals into vectors $\mu \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^n$ and the costs into the matrix $C_{ij} = c(x_i, y_j)$. A coupling is a matrix $\Pi \in \mathbb{R}^{m \times n}$ and Equation 1.26 reads

$$\min_{\Pi \ge 0} \langle C, \Pi \rangle = \sum_{i,j} C_{ij}\Pi_{ij} \qquad \text{subject to} \qquad \Pi \mathbf{1}_n = \mu, \quad \Pi^\top \mathbf{1}_m = \nu. \tag{1.43}$$

The feasible set of Equation 1.43 is the transportation polytope of μ and ν: non-empty, compact and convex. It carries mn non-negative variables against only $m+n$ equality constraints, one of which is redundant, since summing either family returns the total mass 1; so $m+n-1$ are independent. That imbalance is the whole of computational optimal transport. The primal holds a variable for every *pair* of points, the dual of Section 1.12 one per point, and once m and n reach the tens of thousands only the second can be stored at all.

The polytope also settles when Monge and Kantorovich agree. Suppose $m = n$ and both marginals are uniform, $\mu = \nu = \frac{1}{n}\mathbf{1}_n$. The feasible set is then $\frac{1}{n}\mathcal{B}_n$, where $\mathcal{B}_n$ is the

Birkhoff polytope of doubly stochastic matrices. By the Birkhoff–von Neumann theorem the extreme points of $\mathcal{B}_n$ are exactly the permutation matrices, and a linear objective on a compact convex polytope attains its minimum at an extreme point. So an optimal plan may be taken to be $\frac{1}{n}P_\sigma$ for a permutation σ — that is, to be induced by a map after all, and Equation 1.27 holds with equality. The relaxation costs nothing for discrete measures with equal atom weights; it earns its keep in the unequal and continuous cases, which are the ones Proposition 1.7 and Example 1.4 describe.

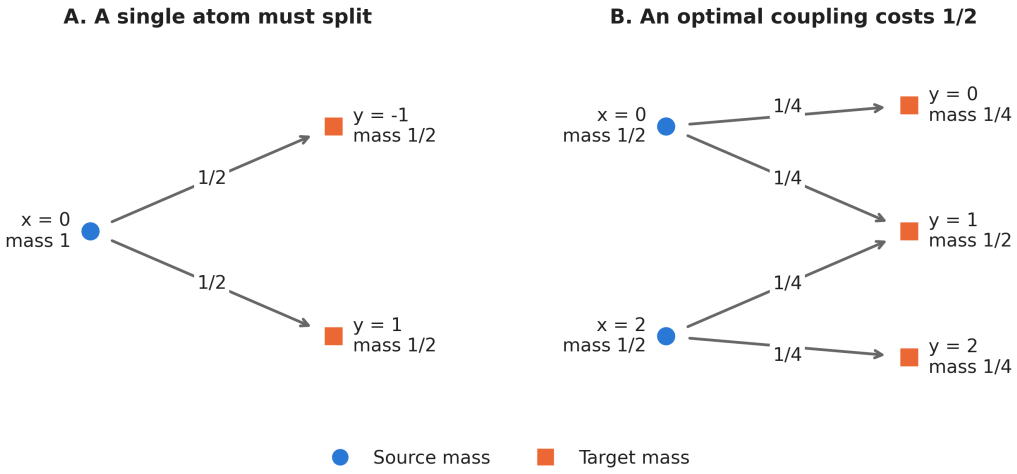


Figure 1.14.: A plan Π drawn as a weighted bipartite graph, which is what Equation 1.43 is. **A.** The obstruction of Proposition 1.7: the single source atom has to send mass to two targets, and only an edge weight can do that. **B.** A plan between μ with mass $\frac{1}{2}$ at 0 and 2 and ν with masses $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ at $0, 1, 2$. Each edge carries one entry Π_{ij}; the edges leaving a source sum to its mass and those entering a target sum to its, which are the two marginal constraints. For $c = |x - y|$ this plan costs $\frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 1 + \frac{1}{4} \cdot 1 + \frac{1}{4} \cdot 0 = \frac{1}{2}$, which is W_1.

1.15.7. Duality beyond finite spaces

The proof of Theorem 1.6 settles strong duality on finite spaces and defers the rest to a citation. Two things that citation hides are worth stating; a consequence for the optimal plans themselves follows in Section 1.15.8.

What the general theorem needs. Strong duality in Equation 1.29 holds whenever $\mathcal{X}$ and $\mathcal{Y}$ are Polish and c is lower semicontinuous and bounded below. Neither continuity of c nor finiteness of the optimal value is required. The proof is not a limit of the finite argument above; it is a Fenchel–Rockafellar duality in the pair $(C_b(\mathcal{X} \times \mathcal{Y}), \mathcal{M}(\mathcal{X} \times \mathcal{Y}))$, with the lower semicontinuous c approached from below by bounded continuous costs.

When the supremum is attained. Equation 1.29 is written with a supremum and not a maximum deliberately: optimal potentials need not exist. They do exist, and are then

called Kantorovich potentials as in Definition 1.14, under a mild domination hypothesis — it is enough that c be continuous and that $c(x,y) \le a(x) + b(y)$ for some $a \in L^1(\mu)$ and $b \in L^1(\nu)$. Both costs used in this chapter satisfy it whenever the measures have the moments Definition 1.10 assumes, since $\|x-y\| \le \|x\| + \|y\|$ and $\frac{1}{2}\|x-y\|^2 \le \|x\|^2 + \|y\|^2$. This is the hypothesis that licenses Theorem 1.8 to begin by taking optimal potentials as given.

1.15.8. Complementary slackness and cyclical monotonicity

Duality constrains the optimal plans themselves and not only the optimal value. The weak-duality computation in Theorem 1.6 integrated an inequality; once the two optimal values agree, that inequality has nowhere left to hide.

Proposition 1.11 (Complementary slackness)**.** *Let π be optimal for Equation 1.26 and (φ, ψ) optimal for Equation 1.29, with the common value finite. Then*

$$\varphi(x) + \psi(y) = c(x,y) \qquad \text{for } \pi\text{-almost every } (x,y), \tag{1.44}$$

so π is concentrated on the set where the dual constraint is active. When $\psi = \varphi^c$, that set is the c-superdifferential of φ.

Proof. The function $c(x,y) - \varphi(x) - \psi(y)$ is non-negative by the definition of Φ_c, and its integral against π is $\mathbb{K}(\pi) - \mathbb{J}(\varphi,\psi)$, which vanishes by Equation 1.29. A non-negative function with zero integral is zero almost everywhere. □

Equation 1.44 is the engine of every structural result about optimal plans; it is what Theorem 1.8 uses to turn a coupling into a map. It also has a consequence that mentions no potentials at all. If π is optimal then $\operatorname{supp}(\pi)$ is **c-cyclically monotone**: for every finite family $(x_1,y_1), \dots, (x_N,y_N)$ in the support and every permutation σ of $\{1,\dots,N\}$,

$$\sum_{i=1}^{N} c(x_i, y_i) \;\le\; \sum_{i=1}^{N} c(x_i, y_{\sigma(i)}). \tag{1.45}$$

The proof is one line from Equation 1.44: summing the equality over the family gives $\sum_i c(x_i,y_i) = \sum_i \varphi(x_i) + \sum_i \psi(y_i)$, the right-hand side is unchanged by reindexing ψ along σ, and $\varphi(x_i) + \psi(y_{\sigma(i)}) \le c(x_i, y_{\sigma(i)})$ term by term. In words: no finite reshuffling of destinations among the pairs the plan actually uses can lower the bill. For c continuous and real-valued the converse holds as well — a coupling with c-cyclically monotone support is optimal — so Equation 1.45 characterises optimality with no reference to the dual problem.

1.15.9. The c-transform for the quadratic cost

For $c(x,y) = \|x-y\|$ the c-transform of a 1-Lipschitz function is simply its negative, which is Lemma 1.3. For the quadratic cost it is the Legendre transform wearing a disguise, and that identification is the whole of the change of variables in Theorem 1.8.

$c(x,y) = \frac{1}{2}\|x-y\|^2$ the c-transform is the Legendre transform wearing a disguise. Put $u(x) = \frac{1}{2}\|x\|^2 - \varphi(x)$ and expand the square:

$$\varphi^c(y) = \inf_x \Big\{ \tfrac{1}{2}\|x-y\|^2 - \varphi(x) \Big\} = \tfrac{1}{2}\|y\|^2 - \sup_x \{\langle x, y\rangle - u(x)\} = \tfrac{1}{2}\|y\|^2 - u^*(y). \quad (1.46)$$

So φ is c-concave for the quadratic cost exactly when $u = \frac{1}{2}\|\cdot\|^2 - \varphi$ is convex and lower semicontinuous: applying Equation 1.46 twice turns c-concavity of φ into the identity $u = u^{**}$, which is the Fenchel–Moreau theorem. It is worth registering that what appears is *convexity* of u and not concavity — the c-transform reverses order, and the $\frac{1}{2}\|\cdot\|^2$ absorbs the sign. This identification is the entire change of variables of Theorem 1.8, which carries it out in a single line.

1.15.10. Continuity of the Wasserstein loss

Example 1.1 shows one generator, with one parameter, on which W_1 carries a gradient and the f-divergences do not. The general statement rests on Section 1.15.4: W_1 metrises weak convergence on a space of bounded diameter, whereas total variation and the Jensen–Shannon divergence induce a strictly finer topology.

Suppose $G_\theta(z)$ is locally Lipschitz in θ, uniformly in z up to an integrable envelope — a condition any feed-forward network with Lipschitz activations satisfies. Then $\theta \mapsto \mu_\theta$ is continuous for weak convergence, so by Equation 1.42

$$\theta \mapsto W_1(\mu_\theta, p_{\text{data}}) \quad (1.47)$$

is continuous everywhere and, being locally Lipschitz, differentiable almost everywhere. Gradient descent on it is therefore well posed wherever it is run.

No such statement is available for D_{JS}. By Theorem 1.2 it is locally constant wherever the supports separate, which by the dimension count of Section 1.7 is essentially everywhere; it is differentiable almost everywhere too, but with derivative zero, which is worse than useless. This is Theorem 1 of Arjovsky, Chintala and Bottou, and it is the reason the loss was worth changing.

1.16. Appendix: proof of the Radon–Nikodym theorem

Theorem 1.1 was used throughout without proof. The following argument, due to von Neumann, is short and uses only the Riesz representation theorem in a Hilbert space, so it is worth including rather than citing.

Proof. Assume first that λ and ν are finite, with $\lambda \ll \nu$, and set $\rho = \lambda + \nu$, also finite. Consider the linear functional

$$L(f) = \int_\Omega f \, d\lambda, \qquad f \in L^2(\rho).$$

It is bounded: by Cauchy–Schwarz in $L^2(\rho)$, and since $\lambda \le \rho$,

$$|L(f)| \le \int |f| \, d\lambda \le \int |f| \, d\rho \le \|f\|_{L^2(\rho)} \, \rho(\Omega)^{1/2}.$$

By the Riesz representation theorem there is $g \in L^2(\rho)$ with

$$\int f \, d\lambda = \int f g \, d\rho = \int f g \, d\lambda + \int f g \, d\nu \qquad \text{for all } f \in L^2(\rho),$$

and rearranging,

$$\int f \, (1 - g) \, d\lambda = \int f g \, d\nu. \tag{1.48}$$

Taking $f = \mathbf{1}_{\{g<0\}}$ in $\int f d\lambda = \int f g \, d\rho$ forces $\rho(\{g < 0\}) = 0$, and taking $f = \mathbf{1}_{\{g>1\}}$ gives $\int_{\{g>1\}} (1-g) \, d\lambda = \int_{\{g>1\}} g \, d\nu \ge 0$ with a strictly negative left-hand side unless $\rho(\{g > 1\}) = 0$. So $0 \le g \le 1$ ρ-almost everywhere.

Let $N = \{g = 1\}$. Taking $f = \mathbf{1}_N$ in Equation 1.48 gives $0 = \nu(N)$, and since $\lambda \ll \nu$ also $\lambda(N) = 0$.

Now fix a measurable A and take $f = \mathbf{1}_A(1 + g + \cdots + g^{n-1})$ in Equation 1.48. The left-hand side telescopes, since $(1 + \cdots + g^{n-1})(1 - g) = 1 - g^n$:

$$\int_A (1 - g^n) d\lambda = \int_A (g + g^2 + \cdots + g^n) d\nu.$$

Let $n \to \infty$. On $\{g < 1\}$ we have $g^n \to 0$, and $\lambda(N) = 0$, so the left side tends to $\lambda(A)$ by dominated convergence. The right side increases to $\int_A \sum_{k \ge 1} g^k \, d\nu = \int_A \frac{g}{1-g} \, d\nu$ by monotone convergence, the integrand being defined ν-almost everywhere because $\nu(N) = 0$. Hence

$$\lambda(A) = \int_A h \, d\nu, \qquad h = \frac{g}{1-g} \geq 0,$$

which is Equation 1.3. For σ-finite measures, partition Ω into countably many sets of finite measure and add the resulting densities.

Uniqueness: if $\int_A h_1 d\nu = \int_A h_2 d\nu$ for every A, then taking $A = \{h_1 > h_2\}$ and $A = \{h_1 < h_2\}$ in turn forces both sets to be ν-null, so $h_1 = h_2$ ν-almost everywhere. □

1.17. Further reading

Full bibliographic details are on the book's References page. The lecture follows Chapters 1 and 13 of the course text; the transport material beyond it is standard and the sources below are where each statement used here is proved in full.

- Ye (2022) — Chapter 1 for metric spaces, convexity and duality; Chapter 13 for statistical distances and optimal transport. Definitions and equation numbers quoted in this chapter follow that text.
- Villani (2003) — Theorem 1.3 is the general form of Theorem 1.6; Chapter 1 covers the c-transform and c-concavity.
- Villani (2009) — Chapter 5 for duality, Chapter 6 for the metric structure of W_p and the gluing lemma.
- Santambrogio (2015) — Theorem 2.9 proves that the monotone map is optimal for costs $h(x-y)$ with h convex, which Theorem 1.4 cites.
- Peyré and Cuturi (2019) — the linear-programming view of Definition 1.13 and its numerics.
- Brenier (1991) — the original form of Theorem 1.8.
- McCann (1995) — removes the moment hypotheses from Brenier's theorem.
- Csiszár (1967) — the origin of Definition 1.6.
- Sriperumbudur et al. (2009) — proves that total variation is the only non-trivial member of both families, the claim made in Section 1.8.
- Gretton et al. (2012) — the source for Proposition 1.6 and for characteristic kernels.
- Goodfellow et al. (2014) — the original adversarial formulation.
- Nowozin et al. (2016) — Table 2 there lists the generators in the convention discussed after Proposition 1.4.
- Arjovsky and Bottou (2017) — the manifold argument of Section 1.7.
- Arjovsky et al. (2017) — Example 1 there is Example 1.1.
- Mroueh et al. (2017) — mean and covariance feature matching.
- Huang and Belongie (2017) — the layer identified with Equation 1.36.

Based on the scribe note of Jungwoo Ahn, with material from Sungwon Yang and Junhyeong Park; edited by Jong Chul Ye.

2. GAN and its Extensions

2.1. Overview

A generative adversarial network (GAN) learns to sample from an unknown data distribution without writing down a likelihood. A generator turns noise into samples, a discriminator tries to tell those samples from real data, and the two are trained against each other. The previous lecture introduced statistical distances and optimal transport; this lecture shows that every GAN variant is, underneath, a choice of statistical distance between the data and the model, together with a way of estimating that distance with a neural network.

The original GAN minimises the Jensen-Shannon divergence (JSD), which gives no useful gradient when the two distributions do not overlap. Two families of fixes follow: f-GAN and LS-GAN generalise the divergence, while W-GAN, McGAN and Geometric GAN replace it by an integral probability metric (IPM). Along the way we meet the non-saturating loss, mode collapse and InfoGAN.

2.2. Notation

Symbol	Meaning
$x \in \mathcal{X} \subseteq \mathbb{R}^d$	a data point
P_r, p_r	data ("real") distribution and its density
$z \sim p_z$	latent noise in $\mathbb{R}^{d_z}$
G_θ, P_θ or P_g, p_g	generator, and the distribution of $G_\theta(z)$
$D(x) \in (0, 1)$	discriminator (probability that x is real)
f, f^*	convex function of an f-divergence and its Fenchel conjugate
$\|f\|_L$	Lipschitz constant of f
$\Pi(P, Q)$	the set of couplings of P and Q
$\sigma(a) = 1/(1 + e^{-a})$	logistic sigmoid

2.3. The GAN minimax game

2.3.1. The objective

Goodfellow et al. (Goodfellow et al. 2014) define a two-player game between a generator G and a discriminator D:

$$\min_G \max_D \; V(D,G) = \mathbb{E}_{x\sim p_r}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D(G(z))\right)\right]. \tag{2.1}$$

V is the log-likelihood of a binary classifier that labels real samples 1 and generated samples 0. The discriminator maximises it and the generator minimises it.

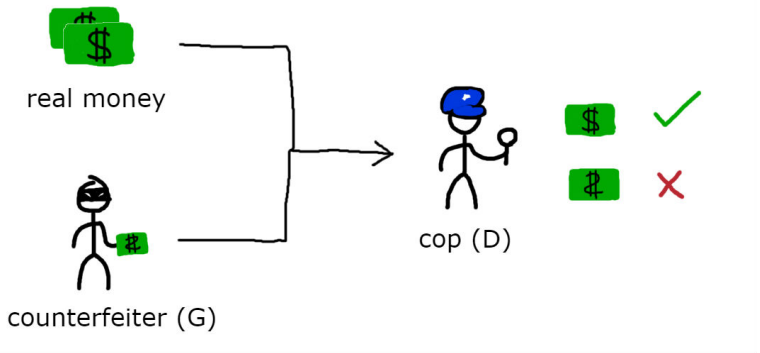


Figure 2.1.: The game of Equation 2.1 before the algebra. A counterfeiter learns to print money that a policeman will accept, and the policeman learns to tell the forgeries from the real notes; neither is given the answer, and each improves only because the other does. The generator is the counterfeiter and the discriminator is the policeman. From the course slides (Ye 2026a).

In practice the game is played by alternating stochastic gradient steps: k ascent steps on the discriminator, then one descent step on the generator. Writing the second expectation over $x = G(z)$,

$$V(D,G) = \int_{\mathcal{X}} \Big(p_r(x)\log D(x) + p_g(x)\log\left(1 - D(x)\right) \Big)\, dx. \tag{2.2}$$

Writing p_g presumes P_g has a Lebesgue density, and Section 2.4 argues the opposite: P_g lives on a d_z-dimensional set with $d_z \ll d$, so no such density exists in exactly the regime we are about to analyse. Nothing below actually needs one. Take any measure ρ dominating both — the mixture $P_r + P_g$ always does — and read p_r, p_g as the Radon–Nikodym derivatives $dP_r/d\rho$, $dP_g/d\rho$, with dx replaced by $d\rho$. Every step through Equation 2.3 goes through unchanged, because each is pointwise in the pair (p_r, p_g). It is the density *notation* that fails on disjoint supports, not the argument.

2.3.2. The optimal discriminator

Proposition 2.1 (Optimal discriminator)**.** *For a fixed generator, the discriminator that maximises Equation 2.2 is*

$$D_G^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)}$$

on the set where $p_r(x) + p_g(x) > 0$.

Proof. The integrand depends on D only through $y = D(x)$ at the same point, so we maximise pointwise. For $a = p_r(x) \geq 0$ and $b = p_g(x) \geq 0$, not both zero, let $h(y) = a \log y + b \log(1 - y)$ on $(0, 1)$. Then

$$h'(y) = \frac{a}{y} - \frac{b}{1-y}, \qquad h''(y) = -\frac{a}{y^2} - \frac{b}{(1-y)^2} < 0,$$

so h is strictly concave and its stationary point $y = a/(a + b)$ is the maximiser. (If $b = 0$ the supremum is approached as $y \to 1$ but not attained — a first hint that disjoint supports cause trouble.) □

The assumption doing the work is that D ranges over *all* functions. A neural discriminator can only approximate D_G^*.

2.3.3. What the generator minimises

Substitute D_G^* into V and write $m = \frac{1}{2}(p_r + p_g)$ for the mixture. Since $\frac{p_r}{p_r+p_g} = \frac{1}{2} \cdot \frac{p_r}{m}$,

$$\begin{aligned} C(G) := \max_D V(D, G) &= \mathbb{E}_{p_r}\left[\log \frac{p_r}{p_r + p_g}\right] + \mathbb{E}_{p_g}\left[\log \frac{p_g}{p_r + p_g}\right] \\ &= \mathbb{E}_{p_r}\left[\log \frac{p_r}{m}\right] - \log 2 + \mathbb{E}_{p_g}\left[\log \frac{p_g}{m}\right] - \log 2 \\ &= \mathrm{KL}(p_r \,\|\, m) + \mathrm{KL}(p_g \,\|\, m) - \log 4 = 2\,\mathrm{JSD}(p_r, p_g) - \log 4. \end{aligned} \tag{2.3}$$

Theorem 2.1 (Global optimum of the GAN game)**.** $C(G) \geq -\log 4$*, with equality if and only if* $p_g = p_r$*. At that point* $D_G^* \equiv 1/2$.

Proof. JSD ≥ 0 with equality iff $p_r = p_g$, since each KL term is zero iff its argument equals m. Plug into Equation 2.3; when $p_g = p_r$, Proposition 2.1 gives $D^* = 1/2$. □

So in the idealised setting the game has the right fixed point. The rest of the lecture is about why reaching it is hard.

2.4. Why the JSD saturates on disjoint supports

2.4.1. The JSD is constant when supports do not overlap

Suppose P_r and P_g are concentrated on disjoint sets S_r and S_g. On S_r we have $p_g = 0$, so $p_r/m = 2$; on S_g, $p_g/m = 2$. Hence

$$\mathrm{KL}(p_r \| m) = \mathrm{KL}(p_g \| m) = \log 2, \qquad \mathrm{JSD}(p_r, p_g) = \log 2,$$

and $C(G) = 2\log 2 - \log 4 = 0$, *however far apart the two sets are.* A generator one pixel away from the data and one producing pure noise get the same value, and the gradient of C with respect to θ is zero.

This is the typical case, not a corner case (Arjovsky and Bottou 2017). Natural images concentrate near a low-dimensional manifold in $\mathbb{R}^d$, and the generator maps $\mathbb{R}^{d_z}$ into $\mathbb{R}^d$ with $d_z \ll d$, so P_g also lives on a low-dimensional set. Two such sets in general position intersect in a set of measure zero, and then a *perfect discriminator* exists: $D = 1$ on S_r, $D = 0$ on S_g.

2.4.2. What this does to the generator's gradient

Write the discriminator as $D = \sigma(a)$, where $a(x)$ is the logit. By the chain rule the generator's term in Equation 2.1 has gradient

$$\nabla_\theta \log\left(1 - \sigma(a)\right) = \frac{\partial \log(1-\sigma(a))}{\partial a} \nabla_\theta a = -D(G_\theta(z))\, \nabla_\theta a(G_\theta(z)). \tag{2.4}$$

When the discriminator confidently rejects the fakes, $D(G(z)) \to 0$ and the factor $-D(G(z))$ kills the gradient, however informative $\nabla_\theta a$ is. The better the discriminator, the less the generator learns; Arjovsky and Bottou show that the generator gradient goes to zero as D approaches a perfect discriminator [2, Thm. 2.4]. Training therefore has a narrow window: too little discriminator training gives a wrong signal, too much gives none.

2.5. The non-saturating loss

2.5.1. The heuristic

Goodfellow et al. already proposed a fix (Goodfellow et al. 2014): instead of minimising $\mathbb{E}_z[\log(1 - D(G(z)))]$, the generator minimises

$$\mathcal{L}_G^{\mathrm{NS}} = -\,\mathbb{E}_{z\sim p_z}\left[\log D(G(z))\right]. \tag{2.5}$$

Both losses push $D(G(z))$ upwards and share the fixed point $p_g = p_r$, but their gradients differ. With $D = \sigma(a)$,

$$\frac{\partial}{\partial a}\left[-\log\sigma(a)\right] = -\left(1-\sigma(a)\right) \longrightarrow -1 \quad \text{as } D \to 0,$$

so the fakes the discriminator rejects most confidently now receive the *largest* gradient instead of the smallest (Figure 2.2).

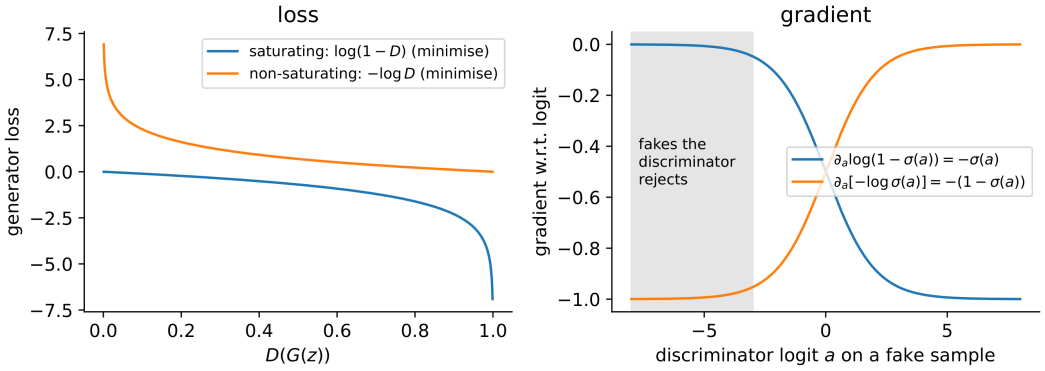


Figure 2.2.: Left: the saturating loss $\log(1-D)$ is flat exactly where the generator is worst ($D(G(z))$ near 0); the non-saturating loss $-\log D$ is steepest there. Right: gradient with respect to the discriminator logit on a fake sample.

2.5.2. What the non-saturating loss actually minimises

The non-saturating loss is no longer the minimax game, so which divergence does it follow? Hold the discriminator at its optimum D^* for the current generator θ_0 and differentiate with respect to θ at θ_0, treating D^* as fixed. Two identities do the work:

$$\log D^* - \log(1-D^*) = \log\frac{p_r}{p_g} \quad\Longrightarrow\quad -\log D^* = \log\frac{p_g}{p_r} - \log(1-D^*),$$

and, from Equation 2.3, $\mathbb{E}_{p_r}[\log D^*] + \mathbb{E}_{p_g}[\log(1-D^*)] = 2\,\mathrm{JSD} - \log 4$, where the first expectation does not depend on θ. Taking $\mathbb{E}_{p_g}$ of the first identity and differentiating gives [2, Thm. 2.5]

$$\nabla_\theta\,\mathbb{E}_z\left[-\log D^*(G_\theta(z))\right] = \nabla_\theta\left[\mathrm{KL}(p_g\|p_r) - 2\,\mathrm{JSD}(p_r,p_g)\right]. \tag{2.6}$$

Two remarks. The $-2\,\mathrm{JSD}$ term has the wrong sign: it pushes the distributions apart. And $\mathrm{KL}(p_g \| p_r)$ is the *reverse* KL: it is infinite if the generator puts mass where there is no data, but costs almost nothing if the generator ignores part of the data. The non-saturating loss buys stable gradients at the price of a bias towards dropping modes.

2.6. Mode collapse

Mode collapse is when the generator covers only a few modes of the data even though each sample may look realistic — for example, one or two of eight Gaussians on a circle, or near-duplicate faces in a batch. A dynamic variant is *mode hopping*: the generator jumps from mode to mode as the discriminator learns to reject the current one.

Three parts of the objective invite it.

1. **The generator's inner problem.** For a *fixed* discriminator, the generator objective $\mathbb{E}_z[-\log D(G(z))]$ is minimised by mapping every z to a single point $\arg\max_x D(x)$. Theorem 2.1 is about $\min_G \max_D$, where the discriminator responds to the generator; alternating gradient steps are closer to alternating best responses, and the generator's best response to a fixed D *is* a point mass.
2. **The divergence being followed.** By Equation 2.6 the non-saturating loss follows the reverse KL, which is mode-seeking: missing a mode is cheap, inventing mass off the data is expensive.
3. **Per-sample judgement.** The discriminator scores samples one at a time, so it cannot notice that a whole batch is the same image.

Remedies include minibatch discrimination (Salimans et al. 2016), unrolled GANs (Metz et al. 2017), and the IPM-based GANs below, whose gradient does not vanish when the discriminator is strong. What the first two actually compute, and how collapse is diagnosed, is in Section 2.18.3.

2.7. f-divergences and f-GAN

2.7.1. f-divergences

Definition 2.1 (f-divergence)**.** Let $f : (0, \infty) \to \mathbb{R}$ be convex and lower semicontinuous with $f(1) = 0$. For distributions $P \ll Q$ with densities p, q,

$$D_f(P\|Q) = \int_{\mathcal{X}} q(x)\, f\!\left(\frac{p(x)}{q(x)}\right) dx.$$

By Jensen's inequality, $D_f(P\|Q) \geq f\left(\int q \cdot \frac{p}{q}\right) = f(1) = 0$. Different choices of f give familiar divergences (Table 2.1).

Table 2.1.: Realisations of the f-GAN, following the lecture (Ye 2026a, 59) and (Nowozin et al. 2016). σ is the logistic function, V_ω the unconstrained critic and $v = V_\omega(x)$; the last two columns are what an implementation needs, since g_f is the output activation that lands V_ω inside $\operatorname{dom} f^*$.

Divergence	$f(u)$	$f^*(t)$	$\operatorname{dom} f^*$	$g_f(v)$	$f^*(g_f(v))$
KL	$u \log u$	e^{t-1}	$\mathbb{R}$	v	e^{v-1}
Reverse KL	$-\log u$	$-1-\log(-t)$	$t < 0$	$-e^{-v}$	$-1+v$
Pearson χ^2	$(u-1)^2$	$t + t^2/4$	$\mathbb{R}$	v	$v + v^2/4$
Squared Hellinger	$(\sqrt{u}-1)^2$	$t/(1-t)$	$t < 1$	$1-e^{-v}$	$e^v - 1$
Jensen–Shannon	$u\log u-(u+1)\log\frac{u+1}{2}$	$-\log(2-e^t)$	$t < \log 2$	$\log 2 - \log(1+e^{-v})$	$-\log(2-2\sigma(v))$
GAN	$u\log u-(u+1)\log(u+1)$	$-\log(1-e^t)$	$t < 0$	$-\log(1+e^{-v})$	$-\log(1-\sigma(v))$

2.7.2. The Fenchel conjugate and the variational lower bound

The *Fenchel conjugate* of f is $f^*(t) = \sup_u\{ut - f(u)\}$. For convex lower semicontinuous f the Fenchel-Moreau theorem gives $f^{**} = f$:

$$f(u) = \sup_{t \in \operatorname{dom} f^*} \{tu - f^*(t)\}. \tag{2.7}$$

To compute a conjugate, solve $f'(u) = t$ for u and substitute. For the GAN row: $f'(u) = \log\frac{2u}{u+1} = t$, which requires $e^t < 2$ and gives $u = e^t/(2-e^t)$. At this u the generator is $f(u) = ut - \log\frac{u+1}{2}$, so $f^*(t) = ut - f(u) = \log\frac{u+1}{2} = -\log(2-e^t)$. For Pearson, $2(u-1) = t$ gives $u = 1 + t/2$ and $f^*(t) = t + t^2/4$.

Now insert Equation 2.7 into Definition 2.1, choosing $t = T(x)$ separately at each point:

$$\begin{aligned} D_f(P\|Q) &= \int q(x) \sup_t \left\{ t\,\frac{p(x)}{q(x)} - f^*(t) \right\} dx \\ &\geq \sup_{T \in \mathcal{T}} \Big(\mathbb{E}_{x\sim P}[T(x)] - \mathbb{E}_{x\sim Q}[f^*(T(x))] \Big). \end{aligned} \tag{2.8}$$

The inequality comes only from restricting T to a class $\mathcal{T}$ (a neural network family); over all measurable T it is an equality (Nguyen et al. 2010), attained at $T^*(x) = f'(p(x)/q(x))$, where the supremum in Equation 2.7 is reached. The right-hand side involves only

expectations, so it can be estimated from samples without knowing either density. That is what makes it usable for a GAN.

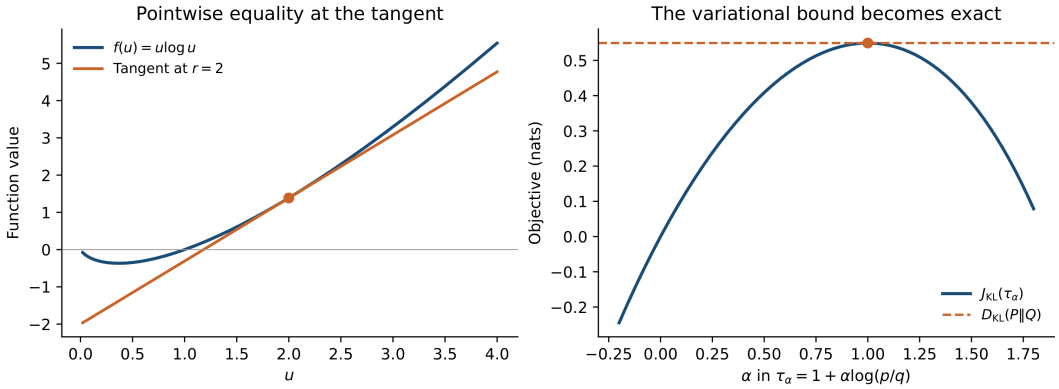


Figure 2.3.: Both halves of the variational bound, for the Kullback–Leibler generator $f(u) = u \log u$. Left: Equation 2.7 at a single point — the supporting line of slope t touches the graph exactly at the density ratio u for which $t = f'(u)$, which is the equality case of the Fenchel–Young inequality and the reason $T^*(x) = f'(p(x)/q(x))$ is optimal. Right: on a two-point example, the objective of Equation 2.8 plotted against a one-parameter family of test functions T_α; it rises to the divergence itself and no further.

2.7.3. f-GAN as the general recipe

Nowozin, Cseke and Tomioka (Nowozin et al. 2016) take $P = P_r$, $Q = P_\theta$ and turn Equation 2.8 into a game:

$$\min_\theta \max_\omega \; F(\theta, \omega) = \mathbb{E}_{x \sim P_r}\left[g_f(V_\omega(x))\right] - \mathbb{E}_{x \sim P_\theta}\left[f^*\big(g_f(V_\omega(x))\big)\right], \tag{2.9}$$

where V_ω is an unconstrained network and g_f is an output activation mapping $\mathbb{R}$ into the domain of f^* (for KL and Pearson, $g_f(v) = v$; for reverse KL, $g_f(v) = -e^{-v}$).

The original GAN, via the normalised generator. Taking the Jensen–Shannon row of Table 2.1, its conjugate needs $t < \log 2$, so take $g_f(v) = \log 2 + \log \sigma(v) = \log\big(2\sigma(v)\big)$. Writing $D(x) = \sigma(V_\omega(x))$ gives $T = \log(2D)$ and $f^*(T) = -\log(2 - 2D) = -\log 2 - \log(1 - D)$, so

$$F(\theta, \omega) = \mathbb{E}_{P_r}[\log D(x)] + \mathbb{E}_{P_\theta}[\log(1 - D(x))] + \log 4 = V(D, G) + \log 4,$$

which is Equation 2.1 shifted by a constant — and it is the same constant that Equation 2.3 carries, since $D_{f_{\mathrm{GAN}}} = 2\,\mathrm{JSD} = C(G) + \log 4$. Maximising over ω and minimising over θ is therefore the same problem as Equation 2.1.

A word on that constant, because it is why Table 2.1 carries two rows that look like the same thing. The **GAN** row uses $u \log u - (u+1)\log(u+1)$, which is what the f-GAN paper and the lecture both use (Ye 2026a, 56–59), and it reproduces Equation 2.1 on the nose: $g_f(v) = -\log(1+e^{-v})$ gives $f^*(g_f(v)) = -\log(1-\sigma(v))$, and the variational objective is exactly $\mathbb{E}_{P_r}[\log D] + \mathbb{E}_{P_\theta}[\log(1-D)]$ with $D = \sigma(V_\omega)$, with no constant to carry. That is its convenience and it is the reason to prefer it when deriving the original GAN.

What it is not is an f-divergence in the sense of Definition 2.1, because $f(1) = -\log 4 \neq 0$. The Jensen argument above does not apply to it, and the quantity it generates is $2\,\mathrm{JSD} - \log 4$, which is negative whenever the two distributions are close. A reader who takes Definition 2.1 seriously would conclude that the GAN divergence is non-negative, and it is not. The **Jensen–Shannon** row is the same generator normalised so that $f(1) = 0$; it costs the $\log 4$ and buys back non-negativity. This is the same normalisation clash as the factor of two in Proposition 1.4 of the previous chapter, and the honest thing is to carry both and say which one is being used.

So the original GAN is not special: *any* f-divergence gives a GAN. The price is that every f-divergence depends only on the density ratio, so it cannot see how far apart two disjoint supports are, and the saturation problem is not solved within this family. How Equation 2.9 is optimised in practice, and what changes when f changes, is in Section 2.18.1.

2.8. LS-GAN and the Pearson chi-squared divergence

2.8.1. The least-squares objective

Mao et al. (Mao et al. 2017) replace the log loss by squared error against target labels a (fake), b (real) and c (what the generator wants the discriminator to say about its samples):

$$\begin{aligned} &\min_D \tfrac{1}{2}\,\mathbb{E}_{x\sim p_r}\left[(D(x)-b)^2\right] + \tfrac{1}{2}\,\mathbb{E}_{z\sim p_z}\left[(D(G(z))-a)^2\right], \\ &\min_G \tfrac{1}{2}\,\mathbb{E}_{z\sim p_z}\left[(D(G(z))-c)^2\right]. \end{aligned} \tag{2.10}$$

Here D has a linear output. With the sigmoid cross-entropy loss, a fake sample already on the "real" side of the decision boundary gives almost no gradient even if it is far from the data. The squared loss keeps penalising it in proportion to its distance from the target, so gradients do not vanish.

2.8.2. The optimal discriminator and the divergence

Pointwise, D minimises $p_r(x)(D-b)^2+p_g(x)(D-a)^2$, a convex quadratic whose minimiser is

$$D^*(x) = \frac{b\,p_r(x) + a\,p_g(x)}{p_r(x) + p_g(x)}.$$

Add to the generator objective the term $\frac{1}{2}\mathbb{E}_{p_r}[(D(x)-c)^2]$, which does not depend on G. With $D = D^*$,

$$2\,C(G) = \int (p_r + p_g)\,(D^* - c)^2\,dx = \int \frac{\big((b-c)\,p_r + (a-c)\,p_g\big)^2}{p_r + p_g}\,dx.$$

Choose the labels so that $b-c=1$ and $b-a=2$ (hence $a-c=-1$), e.g. $a=-1$, $b=1$, $c=0$. The numerator becomes $(p_r-p_g)^2 = \big(2p_g-(p_r+p_g)\big)^2$ and

$$2\,C(G) = \int \frac{\big(2p_g(x) - (p_r(x)+p_g(x))\big)^2}{p_r(x)+p_g(x)}\,dx = \chi^2_{\text{Pearson}}(2p_g \,\|\, p_r + p_g), \tag{2.11}$$

with $\chi^2_{\text{Pearson}}(P\|Q) = \int (p-q)^2/q$, the f-divergence with $f(u) = (u-1)^2$. It is zero iff $p_g = p_r$. (Mao et al. write the arguments in the opposite order; the integral is the same.) The popular choice $a=0$, $b=c=1$ does not satisfy the two conditions but behaves similarly in practice (Mao et al. 2017). LS-GAN therefore minimises a Pearson χ^2-type divergence, which places it in the f-divergence family.

2.9. DCGAN: the architecture that made it train

Everything so far has been about the objective. None of it says what G and D should be, and for the first two years after Equation 2.1 that was the binding constraint: the objective was right and the models would not converge. DCGAN (Radford et al. 2016) is the paper that settled the architecture, and it is worth a section because every model in the rest of this chapter and the next is built on it.

2.9.1. The generator

Figure 2.4 is the whole design. Spatial resolution doubles at each step while the channel count halves, so the tensor is reshaped from deep and small to shallow and large by learned upsampling rather than by interpolation.

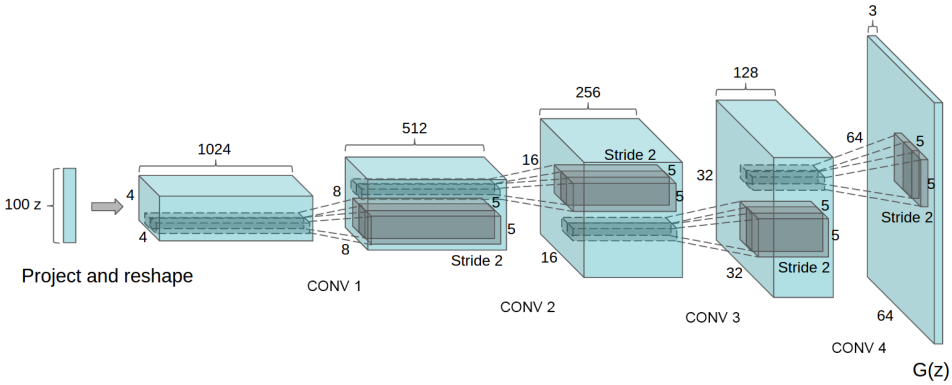


Figure 2.4.: The DCGAN generator. A 100-dimensional uniform z is projected and reshaped to a $4 \times 4 \times 1024$ tensor, then four fractionally-strided convolutions take it to $64 \times 64 \times 3$. There is no fully connected layer after the projection and no pooling anywhere. From (Radford et al. 2016, fig. 1); the lecture slides reproduce it at (Ye 2026a, 64).

2.9.2. Five rules

The paper's contribution is not one architecture but a set of constraints that made the family trainable (Radford et al. 2016):

1. Replace pooling with **strided convolutions** in the discriminator and **fractionally-strided convolutions** in the generator, so the network learns its own downsampling and upsampling.
2. Use **batch normalisation** in both networks.
3. Remove **fully connected hidden layers** in deeper architectures.
4. In the generator use **ReLU** everywhere except the output, which uses **tanh**.
5. In the discriminator use **LeakyReLU** everywhere.

Rules 1 and 3 are the same idea twice: take out the layers that do not respect spatial structure. Rule 2 is the one that interacts with the theory of this chapter. Batch normalisation keeps activations in a range where the discriminator's gradient is informative, which attacks the saturation of Section 2.4 from the architecture side rather than from the loss side — the non-saturating loss of Section 2.5 and the Wasserstein critic of Section 2.11 are the two loss-side attacks on the same problem. Rules 4 and 5 are of a piece with it: a generator ReLU can output zero gradient on a whole half-space, and in the discriminator, where the gradient is the signal passed back to G, the leak is what keeps it alive.

Not every rule survived. Batch normalisation in the discriminator is now usually replaced, because it couples the examples in a minibatch and the gradient penalty of Section 2.12.2 is defined per example; the usual substitutes are layer or instance normalisation, which Section 3.3 of the next chapter treats properly.

2.9.3. What the latent space turned out to be

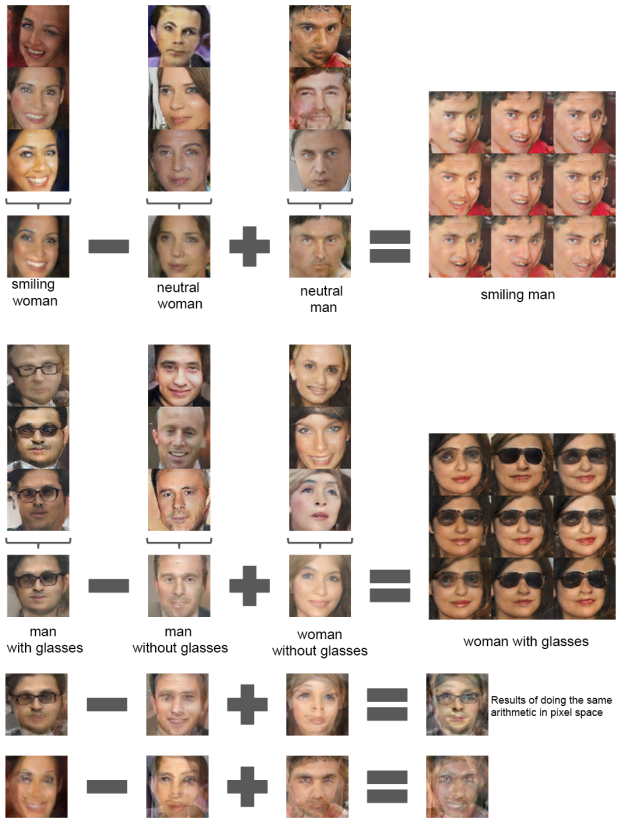


Figure 2.5.: Vector arithmetic in the latent space. In each block, three samples of a visual concept are averaged in z, the averages are combined, and the result is decoded. The bottom two rows do the same arithmetic on pixels instead, which produces superpositions rather than faces. From (Radford et al. 2016, fig. 7); the lecture slides show the first block at (Ye 2026a, 66).

The result in Figure 2.5 was not designed for and is the reason DCGAN is remembered. Averaging the z vectors of smiling women, subtracting the average for neutral women, adding the average for neutral men, and decoding gives smiling men. Nothing in Equation 2.1 asks for this: the objective constrains the push-forward $\mu_\theta = G_{\theta\#}\nu$ as a distribution and says nothing about how G_θ arranges the latent space internally.

The bottom rows are the control, and they are what makes the claim non-trivial. The same arithmetic in pixel space gives ghosted overlays, because pixels have no such structure. So the linear structure is something the generator built, not something the data has.

This is the first appearance of a question the next two chapters are about. If directions in z carry meaning, can they be made to carry *chosen* meanings? InfoGAN below answers

by adding a term that rewards it; Section 3.6 in the next chapter answers by changing where the latent code enters the network; and the β-VAE of Section 4.6 answers by pricing the information the code may carry.

2.10. InfoGAN

2.10.1. The mutual information term

A standard GAN is free to use its latent vector in an entangled way. InfoGAN (Chen et al. 2016) splits the input into noise z and a *latent code* c with a known, factored prior (for MNIST: one 10-way categorical code and two continuous codes, uniform on $[-1, 1]$), and asks that the code be recoverable from the output:

$$\min_G \max_D \; V(D, G) - \lambda \, I(c;\, G(z, c)). \tag{2.12}$$

2.10.2. The variational lower bound

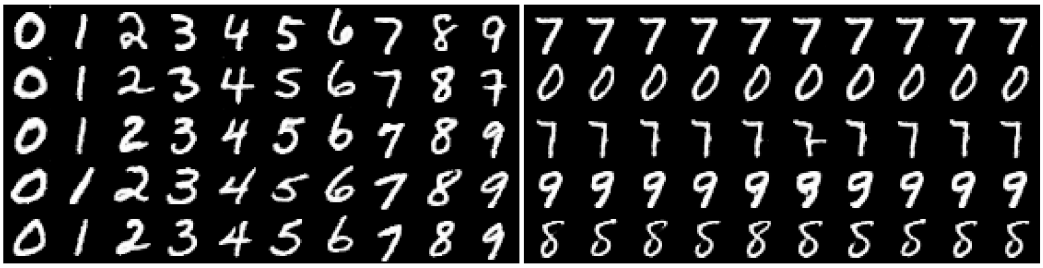
$I(c; x) = H(c) - H(c \mid x)$ needs the posterior $P(c \mid x)$, which is intractable. Introduce an auxiliary distribution $Q(c \mid x)$:

$$\begin{aligned} -H(c \mid x) &= \mathbb{E}_x \, \mathbb{E}_{c' \sim P(c|x)} \left[\log P(c' \mid x) \right] \\ &= \mathbb{E}_x \Big[\mathrm{KL}(P(\cdot \mid x) \,\|\, Q(\cdot \mid x)) + \mathbb{E}_{c' \sim P(c|x)} \left[\log Q(c' \mid x) \right] \Big] \\ &\geq \mathbb{E}_x \, \mathbb{E}_{c' \sim P(c|x)} \left[\log Q(c' \mid x) \right], \end{aligned}$$

because the KL term is non-negative. The remaining expectation still samples from the posterior, but it does not have to: the pair (c', x) with $x \sim P_G$, $c' \sim P(c \mid x)$ has the same joint law as (c, x) with $c \sim p(c)$, $x = G(z, c)$. Hence

$$I(c; G(z, c)) \;\geq\; L_I(G, Q) := \mathbb{E}_{c \sim p(c),\, z \sim p_z} \left[\log Q(c \mid G(z, c)) \right] + H(c), \tag{2.13}$$

which only needs samples from the prior. The bound is tight when Q equals the true posterior, and InfoGAN solves $\min_{G,Q} \max_D V(D, G) - \lambda L_I(G, Q)$. Q is a small head on the discriminator's convolutional trunk, so the extra cost is negligible; for a categorical code $\log Q$ is a cross-entropy, for a continuous code a Gaussian Q gives a squared error.



(a) Varying c_1 on InfoGAN (Digit type) (b) Varying c_1 on regular GAN (No clear meaning)

Figure 2.6.: What the InfoGAN term buys, on MNIST. Each row fixes the noise z and sweeps one latent code across its range. (a) In InfoGAN the categorical code moves along the digit classes, and the sweep is a clean change of one factor. (b) In a plain GAN the same coordinate of the latent vector has no such meaning, and the sweep wanders. This is the latent traversal that Section 2.18.7 describes as the standard of evidence for a disentangled code. From the course slides (Ye 2026a), after Chen et al. (Chen et al. 2016).

2.10.3. What the codes actually disentangle

On MNIST the categorical code aligns with digit class (about 5% error as an unsupervised classifier) and the continuous codes with rotation and stroke width (Chen et al. 2016).

Mutual information is invariant under invertible reparametrisations of c, though, so the objective alone does not decide *which* factor a code captures; that comes from the prior, the architecture and the data. The other datasets, and how λ and the continuous-code head are set, are in Section 2.18.7.

2.11. W-GAN

2.11.1. The Wasserstein-1 distance

From the previous lecture, the Wasserstein-1 (earth mover's) distance is

$$W_1(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right], \tag{2.14}$$

the cheapest cost of moving the mass of P onto Q. Unlike an f-divergence, it looks at *where* the mass is, not only at density ratios: on two disjoint supports W_1 grows with the gap while the JSD is stuck at $\log 2$ (see the worked example).

Theorem 2.2 (Continuity of W1 in the generator parameters)**.** *Let P_θ be the law of $G_\theta(z)$. If $\mathbb{E}_z\|G_\theta(z) - G_{\theta'}(z)\| \to 0$ as $\theta' \to \theta$, then $\theta \mapsto W_1(P_r, P_\theta)$ is continuous. If moreover G_θ is locally Lipschitz in θ with an integrable constant, it is Lipschitz and hence differentiable almost everywhere [9, Thm. 1].*

Proof. Using the same z for both generators defines a coupling of P_θ and $P_{\theta'}$, so $W_1(P_\theta, P_{\theta'}) \leq \mathbb{E}_z\|G_\theta(z) - G_{\theta'}(z)\|$. By the triangle inequality, $|W_1(P_r, P_\theta) - W_1(P_r, P_{\theta'})| \leq W_1(P_\theta, P_{\theta'}) \to 0$. Under the Lipschitz assumption the same bound is at most $\mathbb{E}_z[L(\theta, z)]\,\|\theta - \theta'\|$, and Rademacher's theorem gives differentiability almost everywhere. □

Neither statement holds for the JSD or the KL: in the worked example both jump at $\theta = 0$.

2.11.2. Kantorovich-Rubinstein duality

Equation 2.14 is an infimum over couplings, which we cannot optimise with samples. Its dual is a supremum over functions:

Theorem 2.3 (Kantorovich-Rubinstein duality)**.** *For P, Q with finite first moments,*

$$W_1(P, Q) = \sup_{\|f\|_L \leq 1} \Big(\mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{y\sim Q}[f(y)]\Big).$$

Proof. Weak duality. For any coupling γ and any 1-Lipschitz f,

$$\mathbb{E}_P[f] - \mathbb{E}_Q[f] = \mathbb{E}_{(x,y)\sim\gamma}\left[f(x) - f(y)\right] \leq \mathbb{E}_\gamma\left[\|x - y\|\right].$$

Take the sup over f on the left and the inf over γ on the right.

Why the dual has this form. The general Kantorovich dual is $\sup\{\mathbb{E}_P[\varphi] + \mathbb{E}_Q[\psi] : \varphi(x) + \psi(y) \leq \|x - y\|\}$. For a given φ the best ψ is the c-transform $\varphi^c(y) = \inf_x[\|x - y\| - \varphi(x)]$, which is 1-Lipschitz. Replacing φ by φ^{cc} (also 1-Lipschitz) never lowers the objective, so we may take $\varphi = u$ 1-Lipschitz. For such u, $u^c(y) \leq -u(y)$ (take $x = y$) and $u^c(y) \geq -u(y)$ (since $u(x) - u(y) \leq \|x - y\|$), so $u^c = -u$ and the dual becomes $\sup_{\|u\|_L \leq 1} \mathbb{E}_P[u] - \mathbb{E}_Q[u]$. Equality with the primal is Kantorovich duality [10, Thm. 5.10]. □

2.11.3. The W-GAN objective

Arjovsky, Chintala and Bottou (Arjovsky et al. 2017) parametrise f by a network f_w, the *critic*, constrained to be K-Lipschitz:

$$\min_{\theta} \max_{w:\|f_w\|_L \le K} \mathbb{E}_{x\sim P_r}[f_w(x)] - \mathbb{E}_{z\sim p_z}[f_w(G_\theta(z))]. \tag{2.15}$$

If the family is rich enough, the inner maximum is $K \cdot W_1(P_r, P_\theta)$. There is no log and no sigmoid: the critic outputs a real score, not a probability. Under the assumptions of Theorem 2.2,

$$\nabla_\theta W_1(P_r, P_\theta) = -\,\mathbb{E}_{z\sim p_z}\left[\nabla_\theta f^{\text{opt}}(G_\theta(z))\right],$$

where f^{opt} is an optimal critic. This is the key difference from the original GAN. There, an optimal discriminator gives the gradient of a saturated JSD, which vanishes on disjoint supports (Equation 2.4). Here, an optimal critic gives *exactly* the gradient of W_1, which is informative everywhere. So **the critic may, and should, be trained to optimality**. In practice W-GAN runs 5 critic steps per generator step, and the authors report no mode collapse in their experiments.

2.12. Enforcing the Lipschitz constraint

Equation 2.15 needs $\|f_w\|_L \le K$, which a plain network does not satisfy. Three ways of enforcing it follow.

2.12.1. Weight clipping

The original W-GAN clips every weight to $[-c, c]$ after each update, with $c = 0.01$. The parameter set is then compact, so every f_w is K-Lipschitz for some K. The authors call this "a clearly terrible way" to enforce the constraint (Arjovsky et al. 2017), and Gulrajani et al. (Gulrajani et al. 2017) document why:

- **Capacity underuse.** Most weights end up at $\pm c$, so the critic learns very simple functions.
- **Exploding or vanishing gradients.** Slightly too large a c explodes the gradient, slightly too small makes it vanish.

1 describes the evidence behind both.

2.12.2. WGAN-GP: the gradient penalty

The idea of (Gulrajani et al. 2017) starts from a property of the optimal critic.

Proposition 2.2 (Optimal critics have unit gradient on transport paths)**.** *Let f^{opt} be an optimal critic and γ^* an optimal coupling. Then $f^{\mathrm{opt}}(x) - f^{\mathrm{opt}}(y) = \|x-y\|$ for γ^*-almost every pair, and wherever f^{opt} is differentiable at $x_t = tx + (1-t)y$, $\|\nabla f^{\mathrm{opt}}(x_t)\| = 1$.*

Proof. By strong duality $\mathbb{E}_{\gamma^*}[\|x-y\| - (f^{\mathrm{opt}}(x) - f^{\mathrm{opt}}(y))] = 0$, and the integrand is non-negative by the Lipschitz property, so it vanishes almost surely. A 1-Lipschitz function that rises by exactly $\|x-y\|$ along a segment of that length must rise at unit rate throughout, so its directional derivative along $(x-y)/\|x-y\|$ is 1; since the gradient norm is at most 1, it equals 1. □

So instead of constraining weights, penalise the critic's gradient norm where it matters, on points between real and generated samples:

$$L_{\mathrm{critic}} = \mathbb{E}_{\tilde{x}\sim P_\theta}\left[f_w(\tilde{x})\right] - \mathbb{E}_{x\sim P_r}\left[f_w(x)\right] + \lambda\, \mathbb{E}_{\hat{x}}\left[\left(\|\nabla_{\hat{x}} f_w(\hat{x})\|_2 - 1\right)^2\right], \tag{2.16}$$

with $\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}$, $\epsilon \sim U[0,1]$, and $\lambda = 10$. The penalty is two-sided (it also pushes norms up to 1) and soft, and it is enforced only on the sampled lines, not everywhere. Because it is per-sample, the critic uses layer normalisation instead of batch normalisation.

2.12.3. Spectral normalisation

Miyato et al. (Miyato et al. 2018) bound the Lipschitz constant through the layers. For $f = h_L \circ \cdots \circ h_1$, $\|f\|_L \le \prod_l \|h_l\|_L$. A linear layer $h(x) = Wx + b$ has $\|h\|_L = \sigma_{\max}(W)$, the spectral norm, and ReLU is 1-Lipschitz. Normalising every weight matrix,

$$\bar{W} = \frac{W}{\sigma_{\max}(W)}, \tag{2.17}$$

therefore gives $\|f\|_L \le 1$. Instead of computing $\sigma_{\max}$ exactly, one step of power iteration per update, with a persistent vector u, suffices because W changes slowly:

$$v \leftarrow \frac{W^\top u}{\|W^\top u\|_2}, \qquad u \leftarrow \frac{Wv}{\|Wv\|_2}, \qquad \sigma_{\max}(W) \approx u^\top W v.$$

Unlike clipping it keeps capacity, and unlike the gradient penalty it costs almost nothing. It is usually combined with the hinge loss of Geometric GAN. How tight the bound is, what is done for convolutional layers, and why the normalised layer behaves differently from a clipped one are in Section 2.18.5.

2.13. McGAN: mean and covariance feature matching

2.13.1. Integral probability metrics

W-GAN is one member of a larger family.

Definition 2.2 (Integral probability metric)**.** For a class $\mathcal{F}$ of real functions on $\mathcal{X}$,

$$d_{\mathcal{F}}(P, Q) = \sup_{f \in \mathcal{F}} \Big| \mathbb{E}_P[f(x)] - \mathbb{E}_Q[f(x)] \Big|.$$

The 1-Lipschitz class gives W_1 (Theorem 2.3); $\{\|f\|_\infty \leq 1\}$ gives twice the total variation; the unit ball of a reproducing kernel Hilbert space gives the maximum mean discrepancy. IPMs compare *expectations* of test functions, so they involve no density ratios and stay finite and informative on disjoint supports.

2.13.2. Mean matching

Mroueh, Sercu and Goel (Mroueh et al. 2017) choose functions linear on top of a learned feature map $\Phi_\omega : \mathcal{X} \to \mathbb{R}^m$:

$$\mathcal{F}_{v,\omega} = \left\{ f(x) = \langle v, \Phi_\omega(x) \rangle \ : \ \|v\|_p \leq 1 \right\}.$$

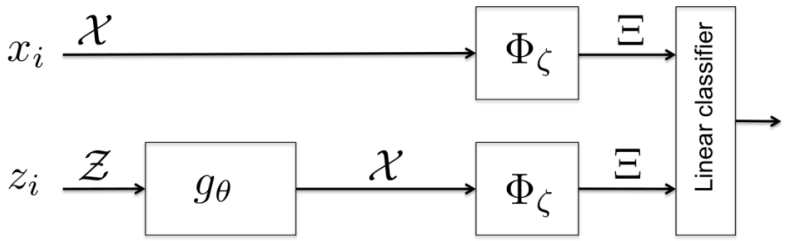


Figure 2.7.: The McGAN critic as a picture. Real samples x_i and generated samples $g_\theta(z_i)$ both pass through the same learned feature map Φ_ζ, and what compares them is a linear classifier on top of those features. The critic's freedom is therefore split in two: the feature map chooses the space, and the linear layer chooses the direction in it. From the course slides (Ye 2026a), after Mroueh et al. (Mroueh et al. 2017).

With $\mu_\omega(P) = \mathbb{E}_{x \sim P}[\Phi_\omega(x)]$, linearity and the definition of the dual norm ($1/p + 1/q = 1$) give

$$d_{\mathcal{F}}(P,Q) = \sup_{\omega} \sup_{\|v\|_p \le 1} \langle v, \mu_\omega(P) - \mu_\omega(Q) \rangle = \sup_{\omega} \|\mu_\omega(P) - \mu_\omega(Q)\|_q. \tag{2.18}$$

The critic finds a feature space in which the two means differ most; the generator matches the means there (Φ_ω is kept bounded by clipping ω).

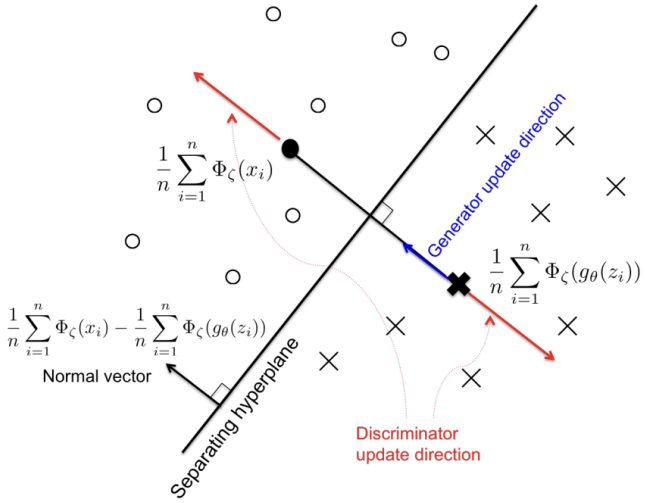


Figure 2: Geometry of the mean feature matching GAN.

Figure 2.8.: Why Equation 2.18 is called mean feature matching. The two clouds are the features of real and generated samples; the filled circle and the cross are their means. The separating hyperplane is normal to the difference of the means, so the discriminator's update rotates that normal to separate the clouds further, and the generator's update pushes its own mean along the normal towards the real one. Both players act on the same vector, from opposite ends. From the course slides (Ye 2026a), after Mroueh et al. [13, Fig. 2].

A W-GAN critic whose last layer is linear, $f(x) = \langle v, \Phi_\omega(x) \rangle$, with v clipped to $[-c, c]$, is exactly mean matching with $p = \infty$, i.e. the ℓ_1 distance between feature means. The two are IPMs of the same form and differ only in the function class.

2.13.3. Covariance matching

Mean matching ignores second-order structure. McGAN adds $f(x) = \langle u, \Phi_\omega(x) \rangle \langle v, \Phi_\omega(x) \rangle = u^\top \Phi_\omega(x) \Phi_\omega(x)^\top v$ for unit vectors u, v. With the uncentred covariance $\Sigma_\omega(P) = \mathbb{E}_P[\Phi_\omega(x)\Phi_\omega(x)^\top]$,

$$\sup_{\|u\|=\|v\|=1} u^\top (\Sigma_\omega(P) - \Sigma_\omega(Q)) v = \sigma_{\max}(\Sigma_\omega(P) - \Sigma_\omega(Q)), \tag{2.19}$$

the largest singular value of the difference. Using k orthonormal pairs instead gives the sum of the top k singular values. The generator then matches the leading directions of the feature covariance (Mroueh et al. 2017). Equation 2.18 and Equation 2.19 are what is being estimated, not what is coded; Section 2.18.2 writes the trained objective down.

2.14. Geometric GAN

2.14.1. The discriminator as a separating hyperplane

Lim and Ye (Lim and Ye 2017) also write the critic as linear in features, $f(x) = \langle w, \Phi_\zeta(x)\rangle + b$, and observe that separating real from fake features is a *linear classification problem in feature space*. The best-understood linear classifier is the support vector machine. The soft-margin SVM on real features (label $+1$) and fake features (label -1) solves

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \max\left(0,\, 1 - f(x_i)\right) + C\sum_j \max\left(0,\, 1 + f(G(z_j))\right). \tag{2.20}$$

The hinge terms are zero for samples beyond the margin on their own side; only samples inside the margin or misclassified (the *support vectors*) contribute.

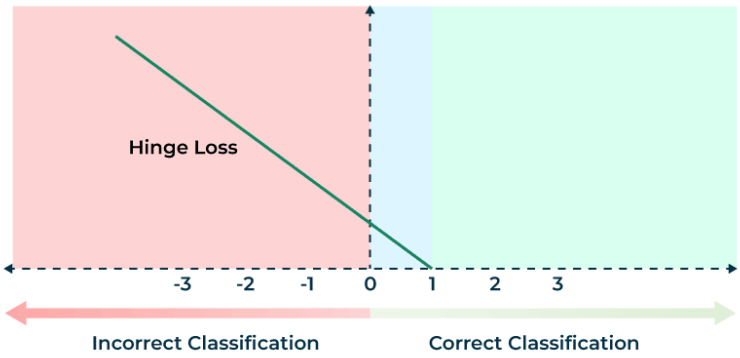


Figure 2.9.: The hinge, drawn. The loss is zero once a sample is correctly classified by a margin of one, and falls linearly as the score moves the wrong way. That flat region is what makes Equation 2.21 depend only on the support vectors: a sample comfortably on its own side contributes no gradient at all. From the course slides (Ye 2026a).

Dropping the $\frac{1}{2}\|w\|^2$ term is not a normalisation identity, and it is worth being explicit about what replaces it: without a bound on $\|w\|$ the hinge loss below can be driven down by rescaling w, and with it the margin interpretation. Lim and Ye instead fix the scale, taking $\|w\| = 1$ or constraining the trunk Φ_ζ — in practice by the spectral normalisation

of Equation 2.17, which is why the two are almost always used together. Under that constraint the regulariser is a constant and the discriminator minimises the hinge loss

$$L_D = \mathbb{E}_{x\sim p_r}\left[\max\left(0, 1 - f(x)\right)\right] + \mathbb{E}_{z\sim p_z}\left[\max\left(0, 1 + f(G(z))\right)\right]. \tag{2.21}$$

2.14.2. What the generator does to the hyperplane

The decision boundary is the hyperplane $\langle w, \phi\rangle + b = 0$ in feature space, with normal vector w. The generator minimises $L_G = -\mathbb{E}_z[f(G(z))]$, whose gradient with respect to a fake sample is $\nabla_x f = J_\Phi(x)^\top w$: the generator moves its features along the normal of the hyperplane, towards the real side. Training alternates between (i) fitting the maximum-margin hyperplane between real and fake features and (ii) pushing the fake features across it. By SVM duality, $w = \sum_i \alpha_i \Phi(x_i) - \sum_j \beta_j \Phi(G(z_j))$ with α_i, β_j nonzero only on support vectors. Compare McGAN, whose direction $\mu_\omega(P) - \mu_\omega(Q)$ averages *all* samples: Geometric GAN uses only the *hard* samples near the boundary. Lim and Ye analyse the equilibrium of the resulting game in (Lim and Ye 2017); Section 2.18.6 states what they prove and what it assumes.

2.14.3. Which distance the hinge discriminator estimates

Pointwise, the discriminator minimises $p_r \max(0, 1 - t) + p_g \max(0, 1 + t)$ over $t = f(x)$. This is piecewise linear with slope $-p_r$ for $t < -1$, $p_g - p_r$ on $[-1, 1]$ and p_g for $t > 1$, so the minimum is at $t = 1$ if $p_r > p_g$ and at $t = -1$ otherwise, with value $2\min(p_r, p_g)$. Using $2\min(a, b) = a + b - |a - b|$,

$$\min_f L_D = \int 2\min\left(p_r, p_g\right) dx = 2 - \int |p_r - p_g|\, dx = 2(1 - \mathrm{TV}(p_r, p_g)).$$

So an unconstrained hinge discriminator measures the total variation distance. With a Lipschitz-constrained critic the loss behaves more like an IPM. The hinge loss with spectral normalisation became the default in later large-scale GANs.

2.15. What the critic's loss does and does not tell you

Standard GAN. The discriminator is never optimal, and the JSD saturates anyway, so its loss is not a measure of sample quality: a falling generator loss can mean a better generator or a weaker discriminator.

W-GAN. If the critic is near-optimal and K-Lipschitz, the negative critic loss (without the penalty) estimates $K \cdot W_1(P_r, P_\theta)$, which is continuous in θ (Theorem 2.2). Arjovsky

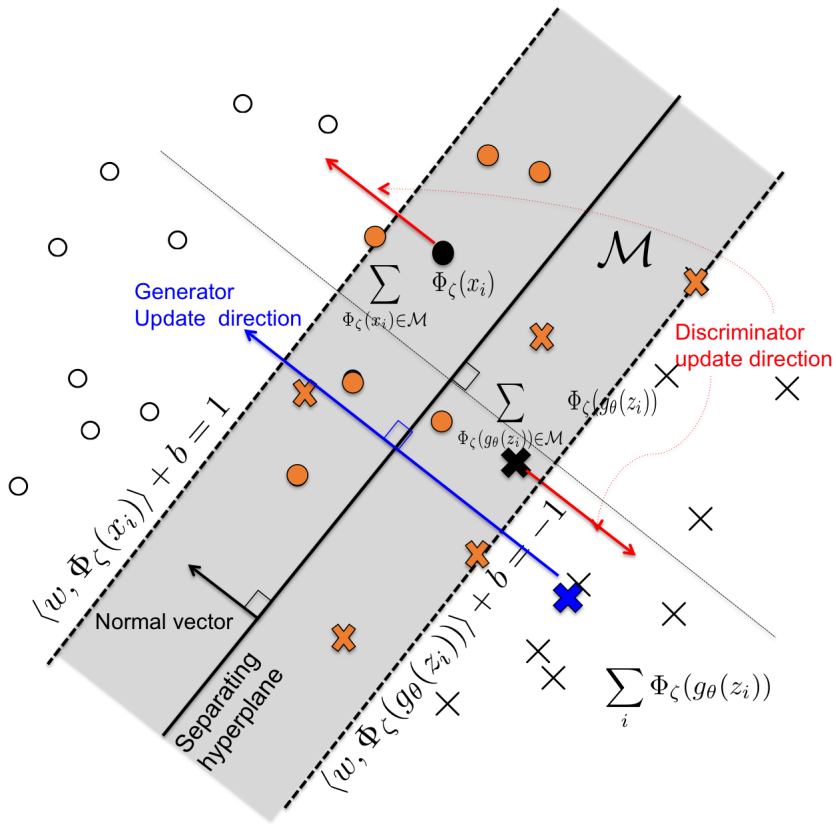


Figure 2.10.: Both players, in feature space. The solid line is the decision boundary $\langle w, \Phi_\zeta(x)\rangle + b = 0$ and the dashed lines the margin at ± 1; circles are real features and crosses generated ones, and the shaded band holds the support vectors $\mathcal{M}$ — the only samples that contribute to Equation 2.21, since the hinge is flat outside it (Figure 2.9). The discriminator's step (red) moves the hyperplane to separate the two clouds; the generator's step (blue) moves its features along the normal w towards the real side, and its direction is set by the sums over $\mathcal{M}$ alone, not over every sample. Compare Figure 2.8, where the same normal is the difference of *all* the feature means. From the course slides (Ye 2026a), after Lim and Ye (Lim and Ye 2017).

et al. show that it decreases steadily as samples improve and correlates with visual quality (Arjovsky et al. 2017), which makes it useful for debugging.

What it does not tell you.

- *It is only a lower bound within the critic's class.* Arora et al. (Arora et al. 2017) show that a generator with small support can nearly fool a finite-capacity critic, so a small value does not certify diversity.
- *Its scale is arbitrary.* It depends on K — the clipping constant, λ or the architecture — so it cannot be compared across models or runs.
- *It is not perceptual.* W_1 with the Euclidean cost measures pixel distances, which do not match human judgement.

In practice samples are also compared with external metrics. The most common, the Frechet Inception Distance (Heusel et al. 2017), $\mathrm{FID} = \|\mu_1 - \mu_2\|_2^2 + \mathrm{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{1/2})$, is the squared Wasserstein-2 distance between Gaussians fitted to Inception features: the same idea as McGAN, but with a fixed rather than learned feature map.

2.16. Worked example: JSD versus W1 on disjoint supports

Let P_0 and P_θ be two narrow Gaussians of width $\sigma = 0.01$ centred at 0 (the data) and at θ (a one-parameter generator). They approximate the point masses δ_0 and δ_θ, whose supports are disjoint for $\theta \neq 0$. For the point masses everything can be computed by hand:

- $\mathrm{JSD}(\delta_0, \delta_\theta) = \log 2 \approx 0.6931$ for every $\theta \neq 0$, so $C(G) = 2\,\mathrm{JSD} - \log 4$ is 0 everywhere except at the optimum, where it jumps to $-\log 4$. Its derivative is zero wherever it exists.
- $W_1(\delta_0, \delta_\theta) = |\theta|$: the only coupling moves all the mass a distance $|\theta|$. Its derivative is $\mathrm{sign}(\theta)$, so a gradient step on θ always moves towards the data.
- An optimal critic is $f^{\mathrm{opt}}(x) = -x\,\mathrm{sign}(\theta)$, whose gradient norm is 1 on the segment, as Proposition 2.2 says.

The code below checks this with the narrow Gaussians, using the fact that in one dimension $W_1(P, Q) = \int |F_P(x) - F_Q(x)|\,dx$ for the cumulative distribution functions.

```
import numpy as np

def densities(theta, sigma, grid):
    p = np.exp(-0.5 * (grid / sigma) ** 2)
    q = np.exp(-0.5 * ((grid - theta) / sigma) ** 2)
    dx = grid[1] - grid[0]
    return p / (p.sum() * dx), q / (q.sum() * dx), dx
```

```python
def jsd(p, q, dx):
    m = 0.5 * (p + q)
    kl = lambda a, b: np.sum(a[a > 0] * np.log(a[a > 0] / b[a > 0])) * dx
    return 0.5 * kl(p, m) + 0.5 * kl(q, m)

def w1(p, q, dx):
    # In one dimension W1 is the L1 distance between the two CDFs.
    return np.sum(np.abs(np.cumsum(p) - np.cumsum(q)) * dx) * dx

grid = np.linspace(-3.0, 5.0, 400001)
for theta in [0.0, 0.01, 0.02, 0.05, 0.1, 0.5, 1.0, 2.0]:
    p, q, dx = densities(theta, 0.01, grid)
    j = jsd(p, q, dx)
    print(f"{theta:.2f} {j:.4f} {2 * j - np.log(4):.4f} {w1(p, q, dx):.4f}")
```

Table 2.2.: JSD and W1 between two Gaussians of width 0.01 separated by θ.

θ	JSD	$2\,\mathrm{JSD} - \log 4$	W_1
0.00	0.0000	-1.3863	0.0000
0.01	0.1114	-1.1635	0.0100
0.02	0.3368	-0.7126	0.0200
0.05	0.6759	-0.0344	0.0500
0.10	0.6931	0.0000	0.1000
0.50	0.6931	0.0000	0.5000
1.00	0.6931	0.0000	1.0000
2.00	0.6931	0.0000	2.0000

Once the two bumps stop overlapping (from about $\theta = 5\sigma = 0.05$), the GAN value function is flat at 0: a generator at $\theta = 0.1$ and one at $\theta = 2$ are indistinguishable to it, and gradient descent has nothing to follow. W_1 equals θ throughout and its gradient always points back to $\theta = 0$ (Figure 2.11). For a real image generator the "width" is zero in most directions of $\mathbb{R}^d$, so the flat regime is the normal one.

2.17. Connections and further reading

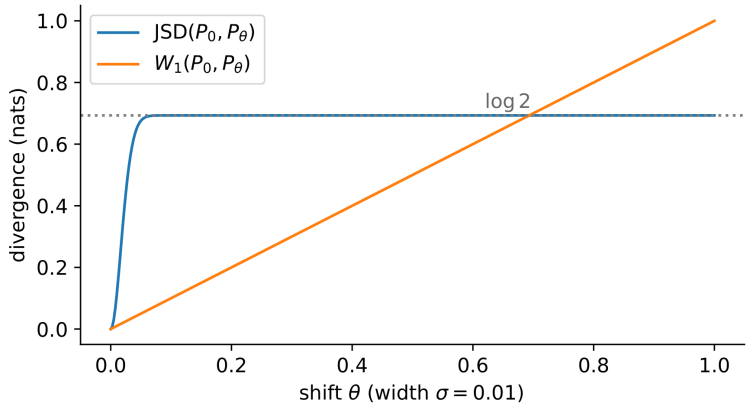


Figure 2.11.: JSD and W1 between two narrow Gaussians as one is shifted by θ. The JSD saturates at $\log 2$ once the supports separate, so its gradient vanishes; W1 grows linearly and always gives a useful gradient.

Table 2.3.: GAN variants and the statistical distance behind each.

Model	Distance (optimal discriminator)	Discriminator output
GAN (Goodfellow et al. 2014)	$2\,\mathrm{JSD} - \log 4$	probability, sigmoid
GAN, non-saturating (Goodfellow et al. 2014; Arjovsky and Bottou 2017)	follows $\mathrm{KL}(p_g \Vert p_r) - 2\,\mathrm{JSD}$	probability, sigmoid
f-GAN (Nowozin et al. 2016)	any D_f, via Equation 2.8	$g_f(V_\omega(x))$
LS-GAN (Mao et al. 2017)	Pearson χ^2, Equation 2.11	linear
W-GAN, WGAN-GP (Arjovsky et al. 2017; Gulrajani et al. 2017)	W_1, an IPM	linear, Lipschitz
McGAN (Mroueh et al. 2017)	IPM on feature means and covariances	linear in features
Geometric GAN (Lim and Ye 2017)	TV if unconstrained; SVM margin	linear, hinge loss

The f-divergence family cannot see how far apart disjoint supports are; the IPM family can, at the cost of constraining the critic. This builds on the optimal transport of Lecture 01, and StyleGAN (Lecture 03) is trained with the non-saturating loss and a gradient penalty on real data.

2.18. Advanced topics

The chapter above follows the lecture. This section collects the material the lecture pointed at without developing: the algorithms behind three of the models, the evidence behind two claims about the Lipschitz constraint, and the equilibrium result for Geometric GAN. Each subsection is self-contained and none is needed to follow the main line.

2.18.1. The f-GAN algorithm in practice

Equation 2.9 is a saddle-point problem, and the obvious way to solve it — an inner loop maximising over ω to convergence, then one step on θ — is expensive and is not what Nowozin, Cseke and Tomioka do. Their *single-step* method takes one ascent step and one descent step per iteration,

$$\omega \leftarrow \omega + \eta \nabla_\omega F(\theta, \omega), \qquad \theta \leftarrow \theta - \eta \nabla_\theta F(\theta, \omega), \tag{2.22}$$

with both gradients taken at the same (θ, ω), so a single pass over a minibatch produces both. The justification is local. Near a saddle point (θ^*, ω^*) at which F is strongly concave in ω and strongly convex in θ, the iteration is a contraction for η small enough, and the authors give the resulting local geometric convergence rate. Neither hypothesis holds globally — GAN objectives are not convex-concave in the parameters — so this is an argument for why the scheme is stable once it is close, not a convergence proof.

Two practical points follow from the recipe rather than from the theory. The output activation g_f is not cosmetic: it is what keeps V_ω inside $\operatorname{dom} f^*$, and choosing it badly makes $f^*(g_f(V_\omega))$ infinite on the first batch. And the choice of f is visible in the samples. Nowozin et al. report that models trained with different divergences from Table 2.1 differ noticeably at the same architecture and budget, in the direction the divergences predict: the mass-covering ones spread over the data and the mode-seeking ones sharpen. This is the same asymmetry as the reverse-KL bias of Equation 2.6, now selected deliberately rather than inherited by accident.

2.18.2. McGAN: the objective as it is trained

Equation 2.18 and Equation 2.19 identify what the mean- and covariance-matching IPMs equal; neither is what a program optimises. The trained problem is a minimax over the feature map and the projection directions together. For the mean term, with Φ_ω the feature map and $\mu_\omega(P) = \mathbb{E}_P[\Phi_\omega]$,

$$\min_\theta \max_{\omega,\, \|v\|_2 \le 1} \big\langle v,\, \mu_\omega(P_r) - \mu_\omega(P_\theta) \big\rangle, \tag{2.23}$$

and for the covariance term, with $U, V \in \mathbb{R}^{m\times k}$ ranging over matrices with orthonormal columns,

$$\min_{\theta} \max_{\omega,\, U^\top U = V^\top V = I_k} \operatorname{tr}\Big(U^\top(\Sigma_\omega(P_r) - \Sigma_\omega(P_\theta))V\Big) \;=\; \min_{\theta}\max_{\omega} \sum_{i\le k} \sigma_i(\Sigma_\omega(P_r) - \Sigma_\omega(P_\theta)), \tag{2.24}$$

the identity being the Ky Fan variational characterisation of the sum of the top k singular values. McGAN trains the sum of the two terms.

Three constraints have to be handled, and each is a line of code rather than a theorem.

- Φ_ω **must be bounded**, or the supremum over ω diverges: a feature map free to scale up makes any non-zero difference of means arbitrarily large. McGAN clamps ω to a box after each update, the same device, with the same drawbacks, as the weight clipping of

  1.

- U **and** V **must stay orthonormal.** They are points on a Stiefel manifold, and an unconstrained gradient step leaves it, so the step is followed by a retraction — in practice a QR or polar factorisation of the updated matrix, keeping the orthonormal factor.
- **The two players alternate.** Ascent on (ω, U, V) and descent on θ, in the same single-step style as Equation 2.22.

Written out this way the relationship to the rest of the chapter is plain. Mean matching with $k = 0$ and a clipped linear last layer is the W-GAN critic of Equation 2.15; replacing the learned Φ_ω by a fixed feature map and taking both terms gives the Frechet Inception Distance; and the whole family sits inside Definition 2.2, differing only in the class $\mathcal{F}$.

2.18.3. Mode collapse: what the standard remedies compute

The chapter names minibatch discrimination and unrolled GANs. Each repairs a different one of the three causes listed above.

Minibatch discrimination answers the third cause, that the discriminator judges samples one at a time. Salimans et al. give the discriminator a feature of the *batch*, not of the sample. Take the trunk's features $\phi(x_i) \in \mathbb{R}^A$ for the n samples in a batch, map each through a learned tensor $T \in \mathbb{R}^{A\times B\times C}$ to a matrix $M_i \in \mathbb{R}^{B\times C}$, and give the discriminator the extra features

$$o(x_i)_b = \sum_{j\neq i} \exp\big(-\|M_{i,b} - M_{j,b}\|_1\big), \qquad b = 1, \dots, B, \tag{2.25}$$

concatenated to $\phi(x_i)$. A batch of near-duplicates makes every $o(x_i)_b$ large, and a diverse batch makes it small, so a collapsed generator is now detectable by a single discriminator evaluation. The discriminator is no longer a function of one sample, which is a real change to the game, but it is a change in exactly the direction the failure points.

Unrolled GANs answer the first cause, that the generator's best response to a *fixed* discriminator is a point mass. Metz et al. let the generator see the discriminator's reply: starting from the current ω, unroll k discriminator ascent steps as a differentiable function $\omega^k(\theta)$ and give the generator the loss $F(\theta, \omega^k(\theta))$, differentiating through the unrolled steps. The extra term $\partial\omega^k/\partial\theta$ is what tells the generator that piling mass on one mode will be punished a few steps later. At $k = 0$ this is the ordinary alternating scheme; k of five to ten is enough to stop the rotation on the eight-Gaussians toy problem, at k times the memory and roughly k times the cost.

Diagnosis is mostly visual, and worth saying plainly because there is no scalar for it. The standard probe is a two-dimensional mixture — eight Gaussians on a circle, or a spiral — where the generator's output can simply be plotted against the target over training: a collapsing generator shows as mass on a subset of the modes, and mode hopping as that subset moving. On images the cheap check is near-duplicate samples within a batch. The critic loss is not a diagnostic: 1 and the discussion of what the critic's loss does not tell you both apply here, and Arora et al.'s point is precisely that a small critic value does not certify diversity.

2.18.4. What weight clipping does to the critic

The two failures the chapter asserts are both documented by Gulrajani et al., and the evidence is worth having because it explains why the gradient penalty is shaped the way it is.

Capacity. A histogram of the critic's weights after training with clipping is bimodal, with almost all the mass at exactly $\pm c$. The clipped critic is therefore close to a sign pattern times c, and the functions it can express are correspondingly crude. On two-dimensional toy distributions the authors plot the value surfaces the critic learns: with clipping these are simple, low-order surfaces that ignore the finer structure of the data, while a gradient-penalised critic follows it. The point is not that the clipped critic is inaccurate by a little, but that it is solving a much smaller problem than the architecture allows.

Gradients. Clipping interacts badly with depth. The critic's gradient scales roughly as a product of per-layer factors, each controlled by c, so the product is exponential in the number of layers: a c slightly too large makes it explode and slightly too small makes it vanish, with the usable window narrowing as the critic gets deeper. Batch normalisation hides this for a while, which is part of why the pathology took a year to be described.

Both observations point the same way: constrain the *function*, not the parameters. That is what Equation 2.16 does by penalising the gradient norm where Proposition 2.2 says it should be one, and what Equation 2.17 does by normalising each layer's spectral norm.

2.18.5. Spectral normalisation: how tight is the bound?

Three caveats belong with Equation 2.17, none of which changes the conclusion that it works, but all of which change what it is claimed to do.

The bound is loose. $\|f\|_L \le \prod_l \sigma_{\max}(W_l)$ holds with equality only if each layer's top singular direction is aligned with the next layer's, which is not what training produces. The true Lipschitz constant of a trained network is typically orders of magnitude below the product, so a spectrally normalised critic is not "1-Lipschitz and no less": it satisfies the constraint with room to spare, and the effective constraint is harder to state than the nominal one.

Convolutions are approximated. $\sigma_{\max}$ in Equation 2.17 is computed by reshaping the kernel $W \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k \times k}$ to a $c_{\text{out}} \times (c_{\text{in}} k^2)$ matrix and taking that matrix's spectral norm. That is not the operator norm of the convolution, which depends on the input resolution, the stride and the padding, and can be larger. The implementation everyone uses is therefore an approximation to an already loose bound. It is a good approximation in practice, and the reason the discrepancy is tolerable is the looseness above.

Why it is not clipping. The interesting difference is in the gradient. Differentiating through $\bar{W} = W/\sigma_{\max}(W)$ gives, for the loss V,

$$\frac{\partial V}{\partial W} = \frac{1}{\sigma_{\max}(W)} \left(\frac{\partial V}{\partial \bar{W}} - \left\langle \frac{\partial V}{\partial \bar{W}}, \bar{W} \right\rangle u_1 v_1^\top \right), \tag{2.26}$$

where u_1, v_1 are the leading singular vectors. The update is the normalised gradient with its component along $u_1 v_1^\top$ removed: whatever the loss would gain by growing the top singular value alone is subtracted out. Miyato et al. read this as a regulariser against the layer collapsing towards rank one. Clipping has no such term — it drives every entry to $\pm c$, which flattens the spectrum by force — and that is the mechanism behind the capacity difference of 1.

2.18.6. Geometric GAN: the equilibrium

The chapter derives the hinge loss and the direction the generator moves, and then defers. What Lim and Ye prove is a statement about the two clouds of features, $\Phi_\zeta(x)$ for $x \sim P_r$ and $\Phi_\zeta(G_\theta(z))$ for $z \sim p_z$.

At each round the discriminator step fits the maximum-margin hyperplane between the two clouds, and the generator step moves the fake features along its normal w towards

the real side. Lim and Ye show that this pair of updates decreases the margin: the geometry of the SVM solution means the displacement is concentrated on the support vectors, which are exactly the fake features nearest the boundary, so the step removes the separation where it is smallest. Iterating, the two clouds converge to each other and the margin collapses; at the equilibrium no hyperplane separates them, which for a linear discriminator on those features is the statement that the two feature distributions coincide.

What the argument assumes is worth naming, because it is where the gap to practice sits. It treats the discriminator step as solving the SVM to optimality rather than taking one gradient step; it holds the feature map Φ_ζ fixed while the hyperplane and the generator move, so it is a statement about the geometry in a *given* feature space and not about learning that space; and it needs the scale fixed, the constraint discussed after Equation 2.20, since otherwise the margin can be changed by rescaling w without moving anything. Matching feature distributions is also weaker than matching the distributions themselves: it is exactly as strong as the feature map is discriminative, which is the same caveat that Definition 2.2 carries for every member of this family.

2.18.7. InfoGAN: what the codes capture

The chapter reports the MNIST result. The paper's case rests on the other datasets, where the codes are less obviously the labels one would have supplied anyway.

Table 2.4.: What InfoGAN's codes align with, by dataset (Chen et al. 2016).

Dataset	Discrete code	Continuous codes
MNIST	digit class, about 5% error	rotation, stroke width
3D faces	—	azimuth, elevation, lighting, width
3D chairs	chair type	rotation, width
CelebA	presence of glasses, hairstyle	azimuth, emotion

The evidence in each row is a *latent traversal*: fix z, sweep one code across its range, and look at the row of images. A code that has captured a factor produces a smooth, monotone change in that factor and leaves the rest alone. This is worth stating as the standard of evidence, because it is the only one available — there is no ground-truth factorisation to score against on CelebA, and the claim being made is about what a human sees in the traversal.

Two settings matter and neither is in Equation 2.12. The weight λ is not shared: the paper uses $\lambda = 1$ for discrete codes and a smaller value for continuous ones, because $\log Q$ is a cross-entropy in the first case and a Gaussian log-density in the second, and the two are on different scales; too large a λ on a continuous code lets the information term dominate and the samples degrade. And for a continuous code the Gaussian head

outputs both a mean and a variance, with the variance *learned* rather than fixed: a fixed variance makes $\log Q$ a plain squared error, and the learned one lets the network express how confident the code is recoverable, which the paper found necessary for the continuous codes to separate.

Based on the scribe note of Sangyun Won, with material from Inhyeok Jeong and Hyungkwon Lee; edited by Jong Chul Ye.

3. StyleGAN, Normalization, Attention, and Image Translation

3.1. Overview

Earlier GAN lectures asked how a discriminator measures mismatch between real and generated distributions. This lecture asks a different question: **where should a generator receive information so that we can control what it generates?** The lecture moves from feature normalization and attention, through StyleGAN and content-style disentanglement, to paired, unpaired, and multi-domain image translation. The organizing equation is $Y = AXT + B$: left multiplication acts on spatial positions, while right multiplication acts on channels.

The main exposition follows *Part 1 - GAN*, pp. 78–125, and the accompanying lecture transcript (Ye 2026a, 2026d). Paper-level equations are added where they clarify a model discussed in class. A section of advanced topics at the end collects the material that was not developed in the provided pp. 78–125/script, so that the main narrative follows the class itself.

Color key. Blue marks spatial/content control; red marks style/channel control; violet marks reconstruction or consistency; teal marks domain supervision; and orange marks diversity. Shared adversarial terms remain black. Labels accompany color so that the comparisons remain readable in grayscale.

3.2. Notation

Symbol	Meaning
$N = HW$, C, M	Spatial positions, feature channels, and minibatch size
$X = [x_1, \dots, x_C] \in \mathbb{R}^{N \times C}$	One image's feature matrix; x_c is a spatial channel map
$x^n \in \mathbb{R}^{1 \times C}$	Row of X: channel features at position n
$\mathbf{1}_N$, I_N, $\mathrm{diag}(v)$	All-ones vector, identity matrix, diagonal matrix

Symbol	Meaning
μ_X, Σ_X	Channel mean vector and centered channel covariance
$x \sim p_{\mathcal{X}}$, $y \sim p_y$	Images from two domains, distinct from feature matrix X
G, F, D	Generators and discriminator; each section states their domains
z, w, s	Prior latent, mapped latent, and style code
k, t, K	Source domain, target domain, number of domains
$T_{\#}p$	Push-forward: the distribution of $T(u)$ when $u \sim p$

Unless stated otherwise, expectations are over the displayed data and independently sampled latent variables. Empirical variances use divisor N, not $N-1$, matching the lecture. Formulas assume positive variances; implementations add $\epsilon > 0$ under square roots. Norm subscripts distinguish the vector L_1/L_2 norms from the matrix Frobenius norm $\|\cdot\|_F$.

3.3. Normalization as channel-wise transformation

3.3.1. What is averaged in batch, layer, and instance normalization?

For a minibatch tensor X_{mnc}, normalization has the common form

$$Y_{mnc} = \gamma \frac{X_{mnc} - \mu}{\sqrt{v + \epsilon}} + \beta. \tag{3.1}$$

The important difference is not this formula, but which entries share its statistics. The following table uses the **CNN layer-normalization convention on slide 87**, which pools spatial positions and channels within one image (Ye 2026a, 83–88; Ioffe and Szegedy 2015; Ba et al. 2016; Ulyanov et al. 2016).

Method	Mean; variance uses the same averaging set	Shared over	Main benefit / limitation
Batch normalization (BN)	$\mu_c = \frac{1}{MN}\sum_{m,n} X_{mnc}$	Batch and positions, separately per channel	Useful optimization conditioning; batch-dependent estimates and train/test statistics can differ
Layer normalization (LN)	$\mu_m = \frac{1}{NC}\sum_{n,c} X_{mnc}$	All features of one image	Independent of batch size; this choice does not standardize each channel separately
Instance normalization (IN)	$\mu_{mc} = \frac{1}{N}\sum_{n} X_{mnc}$	Positions of one image and channel	Removes per-image channel contrast; may discard useful absolute intensity information

For each table row, v is the average of $(X-\mu)^2$ over the indicated indices. BN and IN use channel-specific learned γ_c, β_c. Slide 87 simplifies LN to shared scalar gain and bias; the original LN paper also allows neuron-wise gains and biases. Transformer token-wise LN is another axis choice: it normalizes channels within each token, not all positions in an image. These choices should not be silently interchanged.

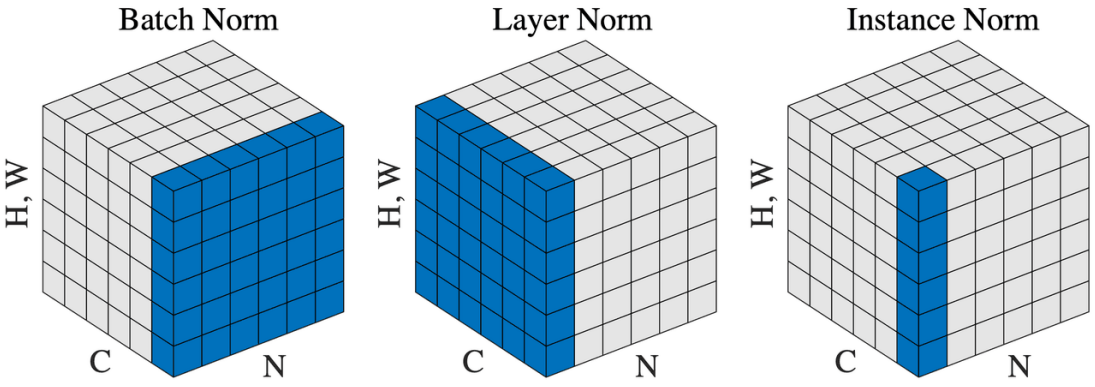


Figure 3.1.: The difference between batch, layer and instance normalization. Each cube is a feature tensor with batch axis N, channel axis C and spatial axes (H, W); the shaded entries are the ones that share a mean and a standard deviation. Adapted from (Wu and He 2018, fig. 2), with the group-normalization panel omitted.

BN uses minibatch statistics during training and population estimates at inference. IN recomputes statistics separately for each image; LN also avoids dependence on other minibatch samples. The transcript initially describes BN statistics as channel-independent but then explicitly corrects this: μ_c and v_c depend on the channel (Ye 2026d).

3.3.2. Deriving the matrix representation

Suppress the batch index. Set $\sigma_c = \sqrt{v_c + \epsilon}$, $\mu = (\mu_1, \dots, \mu_C)^\top$, and $\beta = (\beta_1, \dots, \beta_C)^\top$. Expanding each normalized channel gives

$$\begin{aligned} y_c &= \frac{\gamma_c}{\sigma_c} x_c + \left(\beta_c - \frac{\gamma_c \mu_c}{\sigma_c}\right) \mathbf{1}_N, \\ Y &= X\,T + B, \\ T &= \operatorname{diag}(\gamma_1/\sigma_1, \dots, \gamma_C/\sigma_C), \qquad B = \mathbf{1}_N(\beta^\top - \mu^\top T). \end{aligned} \tag{3.2}$$

This is the lecture's $Y = XT + B$ representation. A diagonal T rescales channels; a dense T can mix them. Neither directly routes a value from one spatial position to another. For IN the matrices depend on the current image; for BN they also depend on the minibatch. Thus this is **an affine representation conditional on the computed statistics**, not a claim that the complete normalization operation is a fixed affine function of X.

3.4. Style transfer: statistics, AdaIN, and transport

3.4.1. Adaptive instance normalization

For content features X and reference-style features S, AdaIN replaces learned fixed IN gains and biases by reference statistics (Huang and Belongie 2017, secs. 3–4; Ye 2026a, 80, 89–90):

$$\begin{aligned} \operatorname{AdaIN}(x_c, s_c) &= \sigma_{S,c} \frac{x_c - \mu_{X,c}\mathbf{1}_N}{\sigma_{X,c}} + \mu_{S,c}\mathbf{1}_N, \\ Y &= (X - \mathbf{1}_N \mu_X^\top) D_X^{-1} D_S + \mathbf{1}_N \mu_S^\top, \quad D_X = \operatorname{diag}(\sigma_X). \end{aligned} \tag{3.3}$$

D_S is defined similarly; X and S need not have the same number of positions. A fixed encoder, this transform, and a learned decoder produce a stylized image. The decoder is trained using content-feature and style-statistic losses. Reference statistics are computed from S, not new free parameters trained for every reference image.

Advantage: efficient arbitrary-reference stylization without an eigendecomposition. **Limitation:** channel means and variances do not determine cross-channel covariance

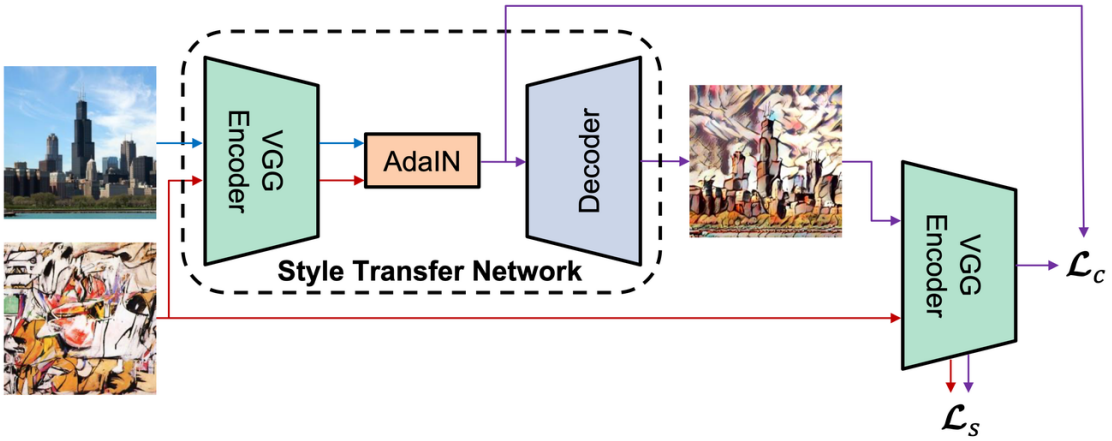


Figure 3.2.: The feed-forward style transfer network of Section 3.11.2. A fixed VGG encoder maps the content and style images to feature space, AdaIN aligns the channel-wise statistics of the content features with those of the style features, and a learned decoder maps the result back to an image. The same encoder computes the content loss $\mathcal{L}_c$ and the style loss $\mathcal{L}_s$ used to train the decoder. Adapted from (Huang and Belongie 2017, fig. 2).

or a general feature distribution. A successful feature transform also need not invert perfectly through the decoder.

3.4.2. Why AdaIN resembles Gaussian optimal transport

The lecture connects AdaIN to Gaussian optimal transport (Ye 2026a, 40, 80; 2026d). Here is the missing one-dimensional derivation. Let $U \sim \mathcal{N}(\mu_u, \sigma_u^2)$ and $V \sim \mathcal{N}(\mu_v, \sigma_v^2)$, with positive standard deviations. For squared Euclidean transport cost, the increasing quantile map is

$$\begin{aligned} \mathcal{T}(u) &= F_V^{-1}(F_U(u)) \\ &= \mu_v + \sigma_v \Phi^{-1}\left(\Phi\left(\frac{u-\mu_u}{\sigma_u}\right)\right) \\ &= \mu_v + \frac{\sigma_v}{\sigma_u}(u-\mu_u). \end{aligned} \tag{3.4}$$

Φ is the standard normal CDF. Consequently, $\mathbb{E}[\mathcal{T}(U)] = \mu_v$ and $\mathrm{Var}[\mathcal{T}(U)] = \sigma_v^2$. For Gaussian channel vectors with **diagonal covariance** and separable squared cost, applying this map coordinate-wise gives AdaIN. The channels may have different means and variances: independence, rather than identical distributions across channels, is the relevant condition.

For positive-definite full covariances, the Gaussian squared-cost map for column vectors is instead

$$\mathcal{T}(u) = \mu_S + M(u - \mu_X), \quad M = \Sigma_X^{-1/2}(\Sigma_X^{1/2}\Sigma_S\Sigma_X^{1/2})^{1/2}\Sigma_X^{-1/2}. \tag{3.5}$$

Thus the lecture's OT interpretation is exact under a Gaussian/diagonal model, not a theorem that empirical CNN features are Gaussian or that AdaIN transports their complete non-Gaussian distributions optimally.

3.4.3. Whitening and coloring transform

WCT extends channel-wise scaling to full channel covariance (Li et al. 2017, sec. 3.2; Ye 2026a, 91). With $X_c = X - \mathbf{1}_N\mu_X^\top$ and $\Sigma_X = X_c^\top X_c/N$, write $\Sigma_X = U_X\Lambda_X U_X^\top$ and similarly for S. In the lecture's row-vector convention,

$$Y = X_c\,\Sigma_X^{-1/2}\Sigma_S^{1/2} + \mathbf{1}_N\mu_S^\top. \tag{3.6}$$

The covariance calculation makes the two steps explicit:

$$\begin{aligned} \mathrm{Cov}(X_c\Sigma_X^{-1/2}) &= I_C, \\ \mathrm{Cov}(Y) &= \Sigma_S^{1/2}\Sigma_X^{-1/2}\Sigma_X\Sigma_X^{-1/2}\Sigma_S^{1/2} = \Sigma_S. \end{aligned} \tag{3.7}$$

Advantage: matches correlations that diagonal AdaIN ignores. **Limitation:** forming a covariance costs $O(NC^2)$ and a dense eigendecomposition costs $O(C^3)$; small eigenvalues need regularization or truncation. Exact full-rank matching assumes positive-definite covariances.

Clarification of the slide: the covariance is centered and the mean correction is $B = \mathbf{1}_N\mu_S^\top - \mathbf{1}_N\mu_X^\top T$, not a channel-wise diagonal expression copied unchanged from AdaIN. WCT is a valid covariance-matching transport, but it is generally not the squared-cost optimal map in Equation 3.5. They coincide in the diagonal, or more generally commuting-covariance, case.

3.5. Attention: transformation along the other axis

3.5.1. Queries, keys, and values

A spatial attention layer combines rows rather than columns. A query represents what an output position seeks, a key supplies matching information, and a value is the information

to be aggregated. With $W_Q, W_K \in \mathbb{R}^{C\times d}$ and $W_V \in \mathbb{R}^{C\times d_v}$, set $Q = XW_Q$, $K = XW_K$, and $V = XW_V$. The output is (Ye 2026a, 92–95; Vaswani et al. 2017)

$$\begin{aligned} a_{mn} &= \frac{\exp(\text{score}(q^m, k^n))}{\sum_{j=1}^{N} \exp(\text{score}(q^m, k^j))}, \\ y^m &= \sum_{n=1}^{N} a_{mn} v^n, \qquad Y = AXW_V. \end{aligned} \tag{3.8}$$

The lecture lists dot-product, scaled dot-product, and cosine scores:

$$qk^\top, \qquad \frac{qk^\top}{\sqrt{d}}, \qquad \frac{qk^\top}{\|q\|_2 \|k\|_2}.$$

For a fixed query index m, $\sum_n a_{mn} = 1$: **rows of A sum to one**. The exponential normalization is softmax, not the sigmoid mentioned hesitantly in the transcript. W_Q, W_K, W_V are learned linear projections; the attention layer becomes nonlinear through its data-dependent weights. In cross-attention, queries and key/value features can come from different inputs, giving a generally rectangular A.

Advantage: dense attention provides direct long-range interaction. **Limitation:** explicitly forming self-attention requires $O(N^2)$ weights; that cost grows rapidly with image resolution. The value projection changes the channel representation, but does not itself standardize a mean or variance.

3.5.2. The unified transform and its limits

The lecture collects these operations as

$$\boxed{Y = \underbrace{A}_{\text{spatial axis}} X \underbrace{T}_{\text{channel axis}} + B.} \tag{3.9}$$

At index level, $Y_{nc} = \sum_{m,j} A_{nm} X_{mj} T_{jc} + B_{nc}$. Reading the two sums separately is what distinguishes the mechanisms. The sum over m runs over input *positions*, so a dense A gathers many positions into one output position while a diagonal A only gates each position in place. The sum over j runs over input *channels*, so a dense T mixes channels while a diagonal T rescales each one across every position at once.

The point of writing it this way is that the operations of this chapter are not four unrelated mechanisms but four choices of which factor is active and where it is computed from (Ye 2026a, 96–97):

Table 3.1.: The operations of this chapter as instances of Equation 3.9.

Operation	Form	A	T	computed from
Instance norm, AdaIN, WCT	$Y = XT + B$	I	channel mixing or scaling	X, and a style input for AdaIN and WCT
Self-attention	$Y = AX$	dense, position mixing	I	X alone
Cross-attention	$Y = AX$	dense, generally rectangular	I	queries from X, keys and values from another input
Channel attention (SENet)	$Y = XT$	I	diagonal	X, pooled over position

Two rows in Table 3.1 are worth separating because they are easy to conflate. Channel attention (Hu et al. 2018) and AdaIN both act through a diagonal T, but AdaIN's scale comes from a *style image* and SENet's comes from the same feature map pooled over position — the first imports information, the second redistributes it. And self- and cross-attention differ only in where the keys come from, which is the whole of the step from a generator that stylises one image to the translation models of Section 3.7.

The lecturer's word *dual* describes complementary tensor axes, not optimization duality. For fixed T, $\mathrm{col}(XT) \subseteq \mathrm{col}(X)$; for fixed A, $\mathrm{row}(AX) \subseteq \mathrm{row}(X)$. However, statistics-dependent A, T, B must be recomputed when their inputs change. Associativity $(AX)T = A(XT)$ for fixed matrices does **not** imply that complete normalization and attention layers commute, or that separate axes guarantee independent semantic factors.

3.6. Generators with style and content control

Here *disentanglement* means being able to vary one designated group of attributes while holding another approximately fixed. The lecture associates content with pose, geometry, and expression, and style with appearance and sometimes identity (Ye 2026a, 104–13). This is an operational convention, not a universal taxonomy.

3.6.1. StyleGAN: learned styles and stochastic detail

StyleGAN starts from a **learned** $4 \times 4 \times 512$ constant tensor shared across generated samples. An eight-layer mapping MLP produces $w = f(z) \in \mathcal{W}$ from $z \sim \mathcal{N}(0, I)$;

layer-specific affine maps produce style parameters (Karras et al. 2019, secs. 2–3; Ye 2026a, 79, 101–3). A synthesis stage can be summarized as

$$\begin{aligned} h_\ell &= \mathrm{Conv}_\ell(\mathrm{Up}(X_{\ell-1})), \\ u_{\ell c} &= \rho(h_{\ell c} + b_{\ell c} n_\ell + a_{\ell c}), \\ X_{\ell c} &= \gamma_{\ell c}(w)\frac{u_{\ell c} - \mu(u_{\ell c})}{\sigma(u_{\ell c})} + \beta_{\ell c}(w), \qquad (\gamma_\ell, \beta_\ell) = L_\ell(w). \end{aligned} \tag{3.10}$$

Here n_ℓ is a spatial Gaussian noise map shared across channels with learned channel scales $b_{\ell c}$; ρ is the nonlinearity. Upsampling is omitted at layers that retain resolution. Unlike reference-based AdaIN, style parameters are predicted from w; a predicted gain need not be a positive reference standard deviation.

Why use $\mathcal{W}$? The mapping can reshape the simple prior before synthesis, rather than forcing all semantic variation to follow its geometry directly. The paper measures improved disentanglement using perceptual path length and linear separability; this is empirical, not guaranteed. Style mixing uses different w values before and after a selected layer. Resampling noise mainly changes stochastic detail, not a controlled semantic attribute.

Advantage: scale-specific control with high-resolution synthesis. **Limitation:** styles can still change pose and geometry, so *style* is broader than texture; complete content-style separation is absent. Its contribution is architectural, not a new GAN divergence.

3.6.2. Structured noise injection: structured content at the input

SNI replaces the fixed input by a tensor assembled from location-specific mappings. A schematic form of its local/shared construction is (Alharbi and Wonka 2020, secs. 3–4)

$$X_0^r = P_r[z_r^{\mathrm{local}}; z_{g(r)}^{\mathrm{region}}; z^{\mathrm{global}}] + b_r, \qquad (\gamma_\ell, \beta_\ell) = L_\ell(f_s(z_s)). \tag{3.11}$$

r indexes input-grid cells and $g(r)$ their shared region. Separate P_r restrict which local latent affects each cell. The paper's preferred configuration uses an 8×8 local grid, shared 2×2 region codes, and one global code; the lecture's 4×4 drawing is a simplified architecture, not its only setting. Style modulation begins later in the preferred configuration.

Advantage: local, regional, and global control without attribute labels. **Limitation:** content enters through the initial grid, while style can enter at many layers. Spatial leakage and remaining entanglement depend on grid/code design; changing background style may also change the foreground. The lecture calls this asymmetry the motivation for hierarchical DAT (Ye 2026a, 105).

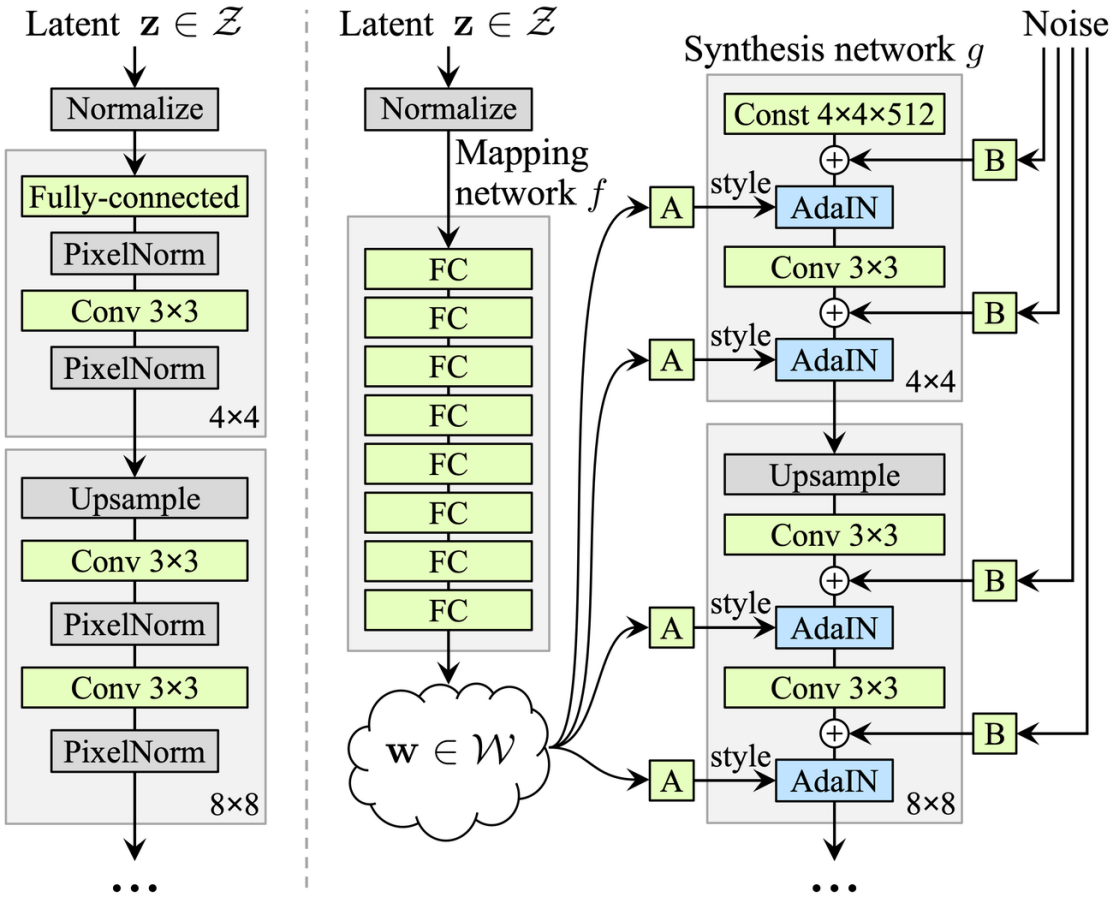


Figure 3.3.: A traditional generator (left) feeds the latent code z through its input layer only. The style-based generator of StyleGAN (right) maps z to w with the mapping network f, feeds w to every AdaIN layer of the synthesis network g through a learned affine transform A, adds per-layer noise scaled by B, and starts the synthesis from a learned constant. Adapted from (Karras et al. 2019, fig. 1).

3.6.3. Diagonal GAN: hierarchical spatial gating

Diagonal GAN introduces content control at several resolutions alongside style control (Kwon and Ye 2021, secs. 3–4; Ye 2026a, 104–13). With independent content and style latents, the paper's residual diagonal attention is

$$\begin{aligned} c = f_c(z_c), \quad s = f_s(z_s), \quad d_\ell = \mathrm{sigmoid}(M_\ell(c)), \\ A_\ell(c) = I + \eta_\ell \,\mathrm{diag}(d_\ell), \\ \widetilde{X}_\ell = A_\ell(c) X_\ell, \qquad Y_\ell = \mathrm{AdaIN}(\widetilde{X}_\ell, s). \end{aligned} \tag{3.12}$$

η_ℓ denotes the paper's learned coefficient β, renamed to avoid confusing it with AdaIN's shift. This is not dense query-key attention: every channel receives the same spatial gate. AdaIN's statistics are computed on its actual incoming tensor $\widetilde{X}_\ell$.

The paper also adds a diversity-sensitive term, often omitted by the slides:

$$\mathcal{L}_{\mathrm{DS}} = \mathbb{E}\left[\max\{\tau - \|G(z_s, z_c^1) - G(z_s, z_c^2)\|_1, 0\}\right], \qquad \mathcal{L}_G = \mathcal{L}_{G,\mathrm{adv}} + \mathcal{L}_{\mathrm{DS}}. \tag{3.13}$$

The shared style is fixed while content varies. Once the distance exceeds τ, this term stops rewarding further separation; it does **not** enforce a hard upper bound on the distance. Training uses non-saturating adversarial loss and R_1 discriminator regularization.

Advantage: spatial control at coarse and fine resolutions with $O(N)$ gate values. **Limitation:** a diagonal gate does not itself move pixels; geometric changes emerge through subsequent convolutions. Separation is an empirical inductive bias, not a theorem of perfect disentanglement.

The lecture illustrates 4×4 pose/geometry, 8×8 expression, and finer eye/shape changes; these are examples rather than universal layer semantics.

3.6.4. Generator-family comparison

Model	Spatial/content input	Style mechanism	Distinctive change
StyleGAN	Shared learned constant; per-layer noise	Latent-predicted AdaIN	Per-resolution styles separate from stochastic detail
SNI	Local, regional, and global codes build the initial tensor	AdaIN	Structured content enters through the initial spatial tensor

Model	Spatial/content input	Style mechanism	Distinctive change
Diagonal GAN	Content-predicted diagonal gates at multiple resolutions	AdaIN	Hierarchical spatial/content gating at several resolutions

The progression in the lecture is therefore **constant + stochastic detail -> structured input -> hierarchical spatial gating**. The key limitation of ordinary StyleGAN noise is that independent noise mainly changes stochastic local detail rather than providing a stable semantic content control signal (Ye 2026d). SNI introduces structured spatial content but does so asymmetrically at the input; Diagonal GAN makes content control hierarchical, paralleling StyleGAN's multi-resolution style control.

3.7. Paired and unpaired image translation

3.7.1. Pix2Pix: realism plus paired reconstruction

Pix2Pix assumes aligned training pairs (x, y). Its discriminator sees the input together with the proposed output, not the output alone (Isola et al. 2017, sec. 3; Ye 2026a, 114–15):

$$\begin{aligned} \mathcal{L}_{\text{cGAN}}(G, D) &= \mathbb{E}_{x,y} \log D(x, y) + \mathbb{E}_{x,z} \log(1 - D(x, G(x, z))), \\ \mathcal{L}_{\text{pair}} &= \mathbb{E}_{x,y,z} \|y - G(x, z)\|_1, \\ G^* &= \arg\min_G \max_D \left[\mathcal{L}_{\text{cGAN}} + \lambda \mathcal{L}_{\text{pair}}\right]. \end{aligned} \tag{3.14}$$

The adversarial term tests conditional realism; the paired term anchors the output to the correct target. A U-Net generator preserves aligned structure through skip connections, and a PatchGAN discriminator checks local patches. The paper uses a non-saturating generator update in implementation and finds that explicit input noise tends to be ignored; dropout produces only modest variation.

Advantage: a shared conditional framework for many aligned translation tasks. **Limitation:** requires paired supervision and does not adequately represent all plausible outputs. It is **not unsupervised simply because it uses a GAN**.

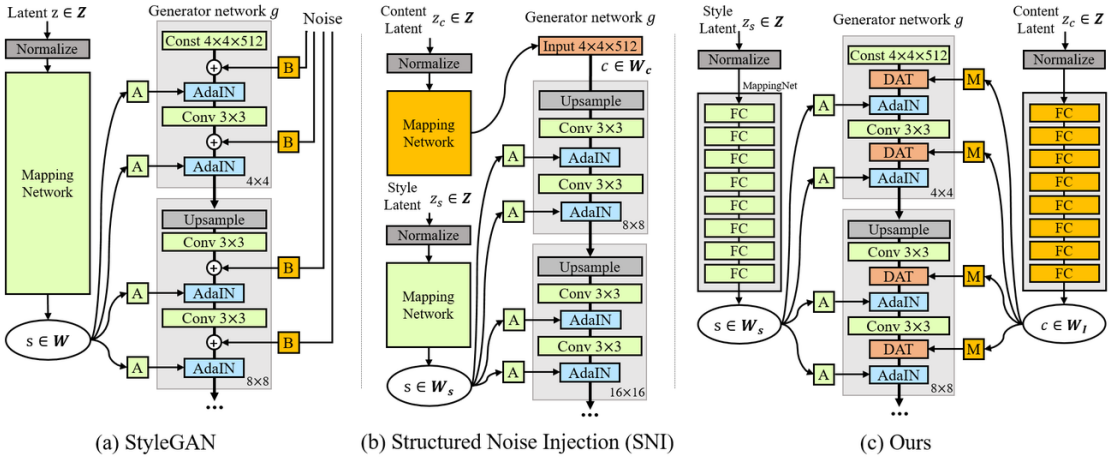


Figure 3.4.: Where content enters, in the three generators of this section. (a) StyleGAN: the style code reaches every layer through the AdaIN boxes, each fed by its own affine transform A, while content arrives only as per-pixel noise scaled by the boxes B. (b) SNI: a separate content code enters once, as the input tensor. (c) Diagonal GAN: the content code reaches every layer as a diagonal attention map, which makes the content path symmetric with the style path. Adapted from (Kwon and Ye 2021, fig. 2).

3.7.2. Why unpaired translation is ill-posed

With unpaired collections, we observe marginals $p_{\mathcal{X}}, p_y$, not the correspondence $p(x, y)$. Matching $G_{\#}p_{\mathcal{X}} = p_y$ does not tell us which target belongs to a particular input. For example, two equally likely inputs and two equally likely targets admit both possible bijections with exactly the same marginals. A constraint is necessary, but a constraint need not identify the intended semantics.

3.7.3. CycleGAN and the role of the cycle

CycleGAN uses $G : \mathcal{X} \to \mathcal{Y}$, $F : \mathcal{Y} \to \mathcal{X}$, and two discriminators (Zhu et al. 2017, sec. 3; Ye 2026a, 116–18):

$$\begin{aligned} \mathcal{L}_{\mathrm{GAN}}(G, D_Y) &= \mathbb{E}_y \log D_Y(y) + \mathbb{E}_x \log(1 - D_Y(G(x))), \\ \mathcal{L}_{\mathrm{cyc}} &= \mathbb{E}_x \|F(G(x)) - x\|_1 + \mathbb{E}_y \|G(F(y)) - y\|_1, \\ \min_{G,F} \max_{D_X, D_Y} & \quad \mathcal{L}_{\mathrm{GAN}}(G, D_Y) + \mathcal{L}_{\mathrm{GAN}}(F, D_X) + \lambda_{\mathrm{cyc}} \mathcal{L}_{\mathrm{cyc}}. \end{aligned} \tag{3.15}$$

The reverse adversarial term exchanges x, y and F, G. The cycle compares a reconstructed image to its own input, so it needs no paired target in the other domain. The published

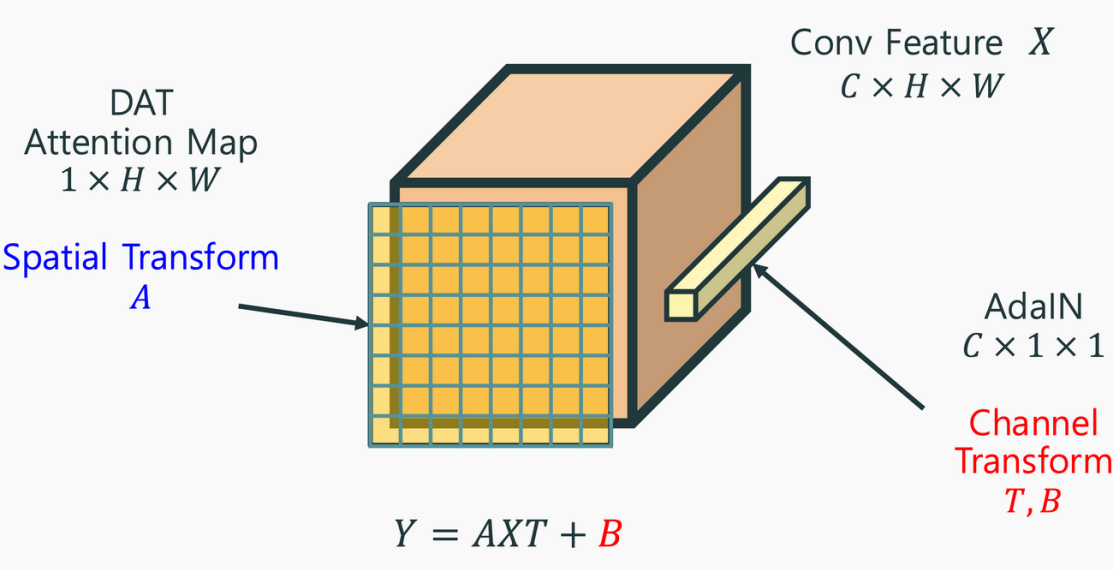


Figure 3.5.: Where the two transforms act within one layer. The feature map is a cube of size $C \times H \times W$. The AdaIN parameters are a $C \times 1 \times 1$ vector, one scale and one shift per channel, applied identically at every pixel; the DAT attention map is a $1 \times H \times W$ map, one weight per pixel, applied identically to every channel. This is Equation 3.9 read one layer at a time. Adapted from (Ye 2026a).

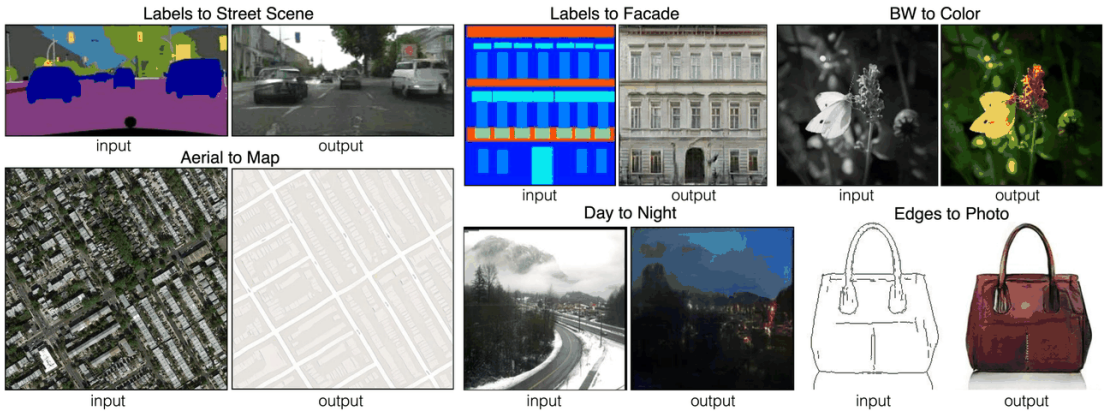


Figure 3.6.: Image-to-image translation as one problem with many instances: in each case the same architecture and objective are trained on different paired data. Adapted from (Isola et al. 2017, fig. 1).

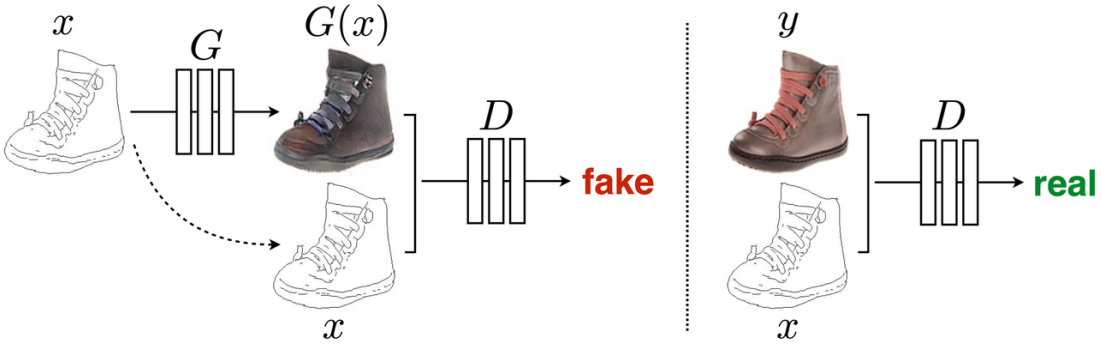


Figure 3.7.: The discriminator of a conditional GAN sees the input together with the output, which is what Equation 3.14 means by $D(x, y)$. Left: a generated pair $(x, G(x))$ should be classified as fake. Right: a real pair (x, y) should be classified as real. Adapted from (Isola et al. 2017, fig. 2).

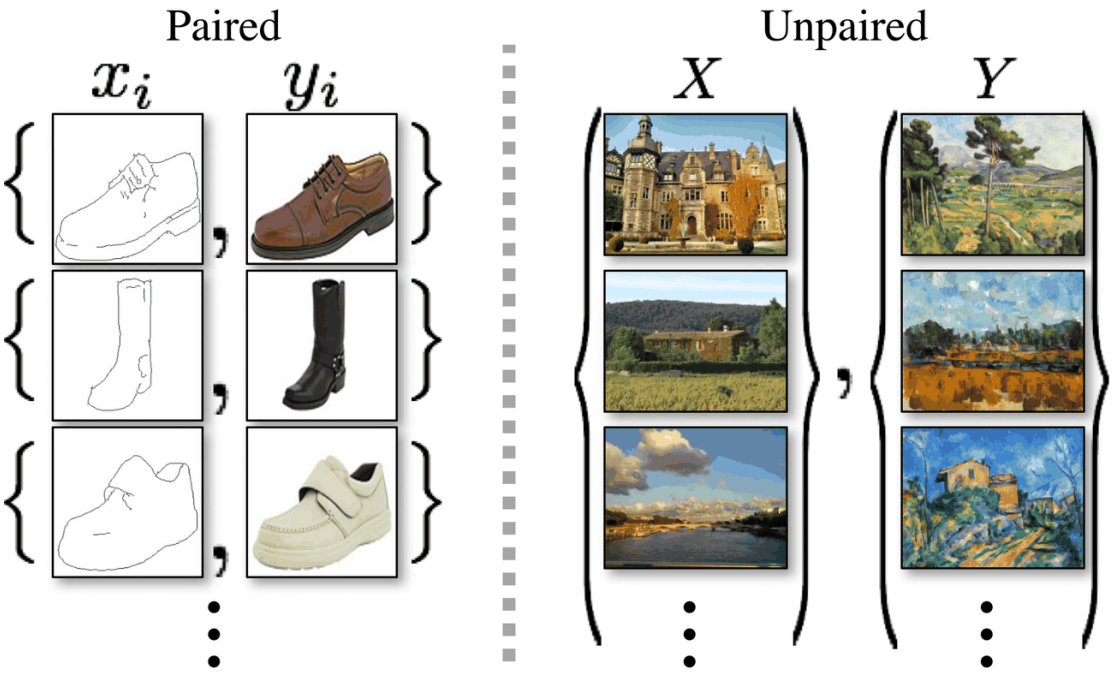


Figure 3.8.: Paired training data (left) consist of examples $\{x_i, y_i\}$ whose correspondence is known. Unpaired data (right) consist of a set X and a set Y with no information about which x matches which y. Adapted from (Zhu et al. 2017, fig. 2).

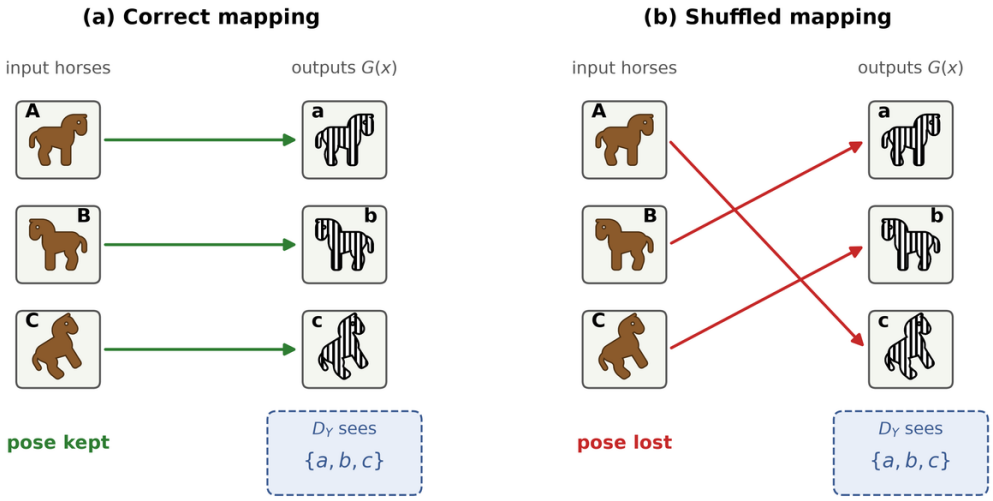


Figure 3.9.: Why matching marginals is not enough, drawn for horses and zebras, with the pose of each animal standing for the content a translation should keep. (a) The correct mapping keeps the pose; (b) a shuffled mapping loses it. Both produce the same set of outputs, so a discriminator that sees only outputs has no reason to prefer (a) — the argument of Section 3.11.6. Horse icon from Font Awesome Free, licensed under CC BY 4.0.

implementation uses least-squares adversarial losses; the equation above is the paper's explanatory log-GAN formulation. An optional identity loss helps preserve appearance when an image is already in the target domain.

Advantage: learns two-way translation from unpaired data.

Limitation: cycle consistency does not establish the correct semantic pairing, and optimization is substantially more delicate than in Pix2Pix. The lecture emphasizes the practical reason: CycleGAN jointly trains two generators and two discriminators, so adversarial objectives in both directions interact with the cycle term. In practice the networks are updated alternately rather than by solving a closed-form minimax problem; poorly balanced updates can produce unstable dynamics. This training difficulty was presented in class as an important hands-on lesson, not a minor implementation detail (Ye 2026d). Figure 3.12 contrasts paired supervision, round-trip reconstruction, domain-conditioned reconstruction, and style-code conditioning across the main translation models.

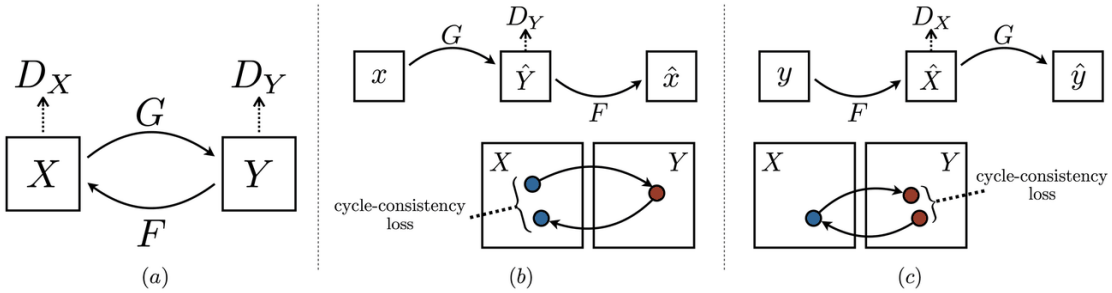


Figure 3.10.: The three parts of Equation 3.15. (a) Two mappings $G : X \to Y$ and $F : Y \to X$, each with its own discriminator. (b) Forward cycle consistency, $x \to G(x) \to F(G(x)) \approx x$. (c) Backward cycle consistency, $y \to F(y) \to G(F(y)) \approx y$. Adapted from (Zhu et al. 2017, fig. 3).

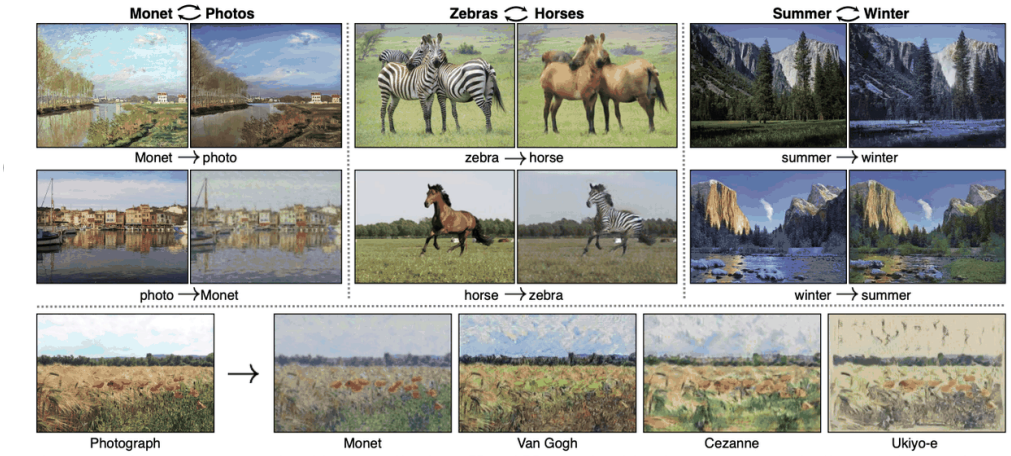


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

Figure 3.11.: What unpaired translation produces, which the rest of this section describes only as an objective. Each pair is a source image and the output of a CycleGAN trained on two unpaired collections: photographs and Monet paintings, horses and zebras, summer and winter. Note what changes and what does not — colour and texture are rewritten throughout, while the layout of the scene and the pose of the animal survive, which is the bias Section 3.11.7 identifies as doing the real identification. Adapted from (Zhu et al. 2017, fig. 1).

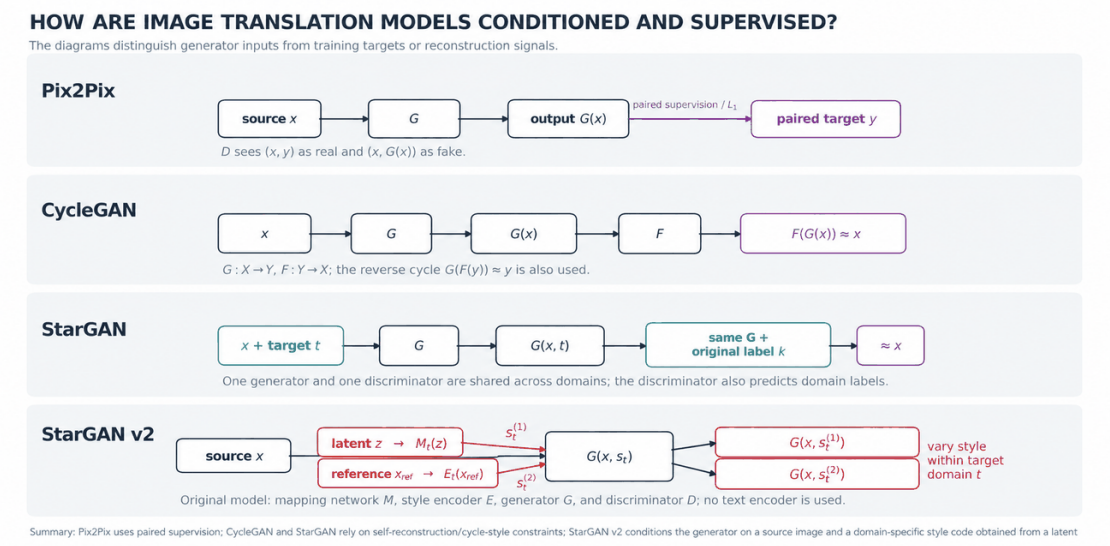


Figure 3.12.: Comparison of conditioning and supervision across image-to-image translation models. Pix2Pix uses a paired target y to supervise $G(x)$; CycleGAN reconstructs the source after the round trip $x \to G(x) \to F(G(x))$; StarGAN reuses the same generator with the original domain label k to reconstruct the source; and StarGAN v2 feeds both a source image x and a target-domain style code s_t, where s_t comes either from the mapping network $M_t(z)$ or the style encoder $E_t(x_{\text{ref}})$.

3.8. Multiple domains and multiple inputs

The lecture presents a clear architectural progression. Pairwise CycleGAN/DiscoGAN-style translation scales poorly when every ordered domain pair needs its own mapping. StarGAN replaces those pairwise generators with one shared generator conditioned on a target domain label. StarGAN v2 then replaces the coarse label-only control with a compact, domain-specific style code that can be obtained from either a random latent or a reference image (Ye 2026a, 119–25; 2026d).

3.8.1. Scaling up: DiscoGAN and RadialGAN

For K domains, separate directed pairwise generators require $K(K-1)$ maps: choose one source and one different target. This is a naive pairwise construction, not a universal lower bound. Slide 119 counts K discriminators by sharing one per destination; independently trained CycleGAN pairs instead contain $K(K-1)$ discriminator instances.

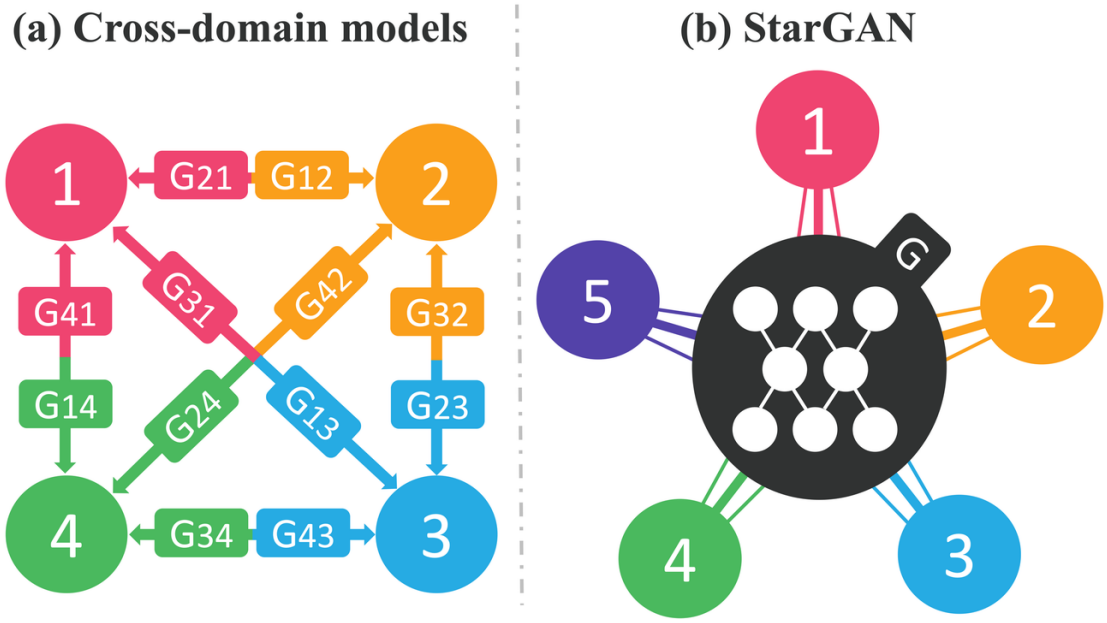


Figure 3.13.: The counting argument of this section, drawn. (a) Translating among four domains with cross-domain models such as CycleGAN needs twelve generators, one for every ordered pair. (b) StarGAN serves five domains with a single generator that takes the target domain as an input. Adapted from (Choi et al. 2018, fig. 2).

DiscoGAN independently develops bidirectional unpaired translation with adversarial losses and two reconstruction cycles (Kim et al. 2017, sec. 2). In the present notation its generator objective has the structure

$$\mathcal{L}_G = \mathcal{L}_{\text{adv}}^{X\to Y} + \mathcal{L}_{\text{adv}}^{Y\to X} + \lambda_X \mathbb{E}_x d(F(G(x)), x) + \lambda_Y \mathbb{E}_y d(G(F(y)), y). \tag{3.16}$$

d is a reconstruction distance; the paper's illustration uses squared error. **Advantage:** discovers cross-domain relations without paired targets. **Limitation:** the basic formulation is two-domain, with the same correspondence ambiguity. It is related to CycleGAN, not necessarily identical in architecture or every loss detail.

RadialGAN routes domains through a shared latent space, with domain-specific encoders E_i and decoders G_j (Yoon et al. 2018, sec. 3):

$$u_i \overset{E_i}{\longrightarrow} z \overset{G_j}{\longrightarrow} \hat{u}_j, \qquad \mathcal{L}_{\text{cyc}}^{i} = \mathbb{E}\|u_i - G_i(E_i(u_i))\|_2 + \mathbb{E}\|z - E_i(G_i(z))\|_2. \tag{3.17}$$

Here u_i denotes a record including its response in the paper's continuous-response setting, and z can come from other domains' encoders. Domain adversarial losses match translated and real records.

Advantage: $O(K)$ encoder/decoder modules accommodate heterogeneous feature spaces. **Limitation:** requires a useful shared representation and still uses multiple domain modules. It is originally a predictive-data transfer method, not a many-image fusion model or a single universal generator like StarGAN.

3.8.2. StarGAN: one generator conditioned on a domain label

StarGAN spatially replicates a target label t and concatenates it with input x. Its discriminator has a real/fake output and an auxiliary domain classifier (Choi et al. 2018, sec. 3; Ye 2026a, 120–22).

$$\begin{aligned}
\mathcal{L}_{\text{cls}}^{r} &= \mathbb{E}_{x,k}[-\log D_{\text{cls}}(k \mid x)], \\
\mathcal{L}_{\text{cls}}^{f} &= \mathbb{E}_{x,t}[-\log D_{\text{cls}}(t \mid G(x,t))], \\
\mathcal{L}_{\text{rec}} &= \mathbb{E}_{x,k,t}\|x - G(G(x,t),k)\|_1, \\
\mathcal{L}_D &= -\mathcal{V}_{\text{adv}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}^{r}, \\
\mathcal{L}_G &= \mathcal{V}_{\text{adv}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}^{f} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}.
\end{aligned} \tag{3.18}$$

$\mathcal{V}_{\text{adv}}$ is the paper's log-GAN value in its explanatory formulation; its experiments replace this with WGAN-GP. Classification on real images trains D; classification on generated images trains G. The same G is used twice for reconstruction. A dataset mask can indicate which labels are known when training jointly on datasets with different annotation sets.

Advantage: shares parameters across domains instead of training every pair separately. **Limitation:** the original deterministic mapping offers one output for fixed (x, t) and needs domain labels. Label concatenation does not provide within-domain style diversity or fuse several source images.

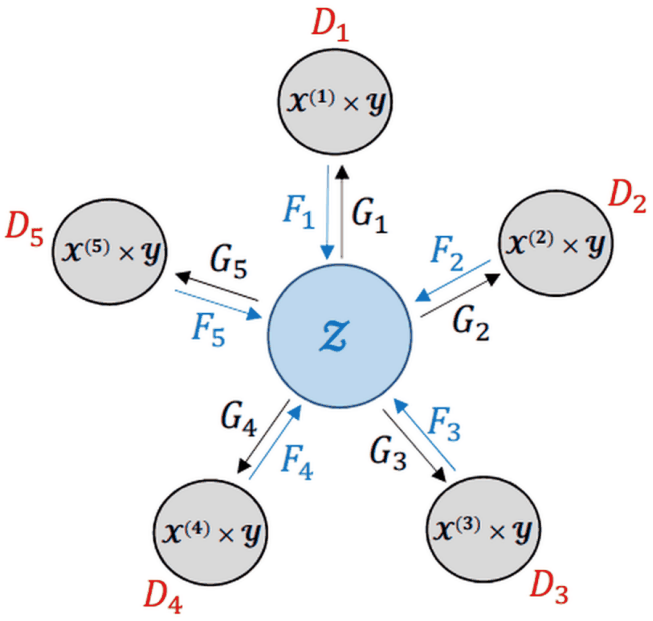


Figure 3.14.: RadialGAN's shape, and the reason for its name. Each outer node is one domain's record space $\mathcal{X}^{(i)} \times \mathcal{Y}$ — features together with the response, since the paper's setting is prediction — and the hub is the shared latent space $\mathcal{Z}$. Every domain reaches every other only through the hub, so adding a domain adds one encoder, one decoder and one discriminator rather than a mapping to each existing domain: $O(K)$ modules where the pairwise construction of this section needs $K(K-1)$. The slide writes F_i for the encoder that Equation 3.17 calls E_i; the black arrows G_i are the decoders and D_i is the discriminator of domain i. From the course slides, after (Yoon et al. 2018).

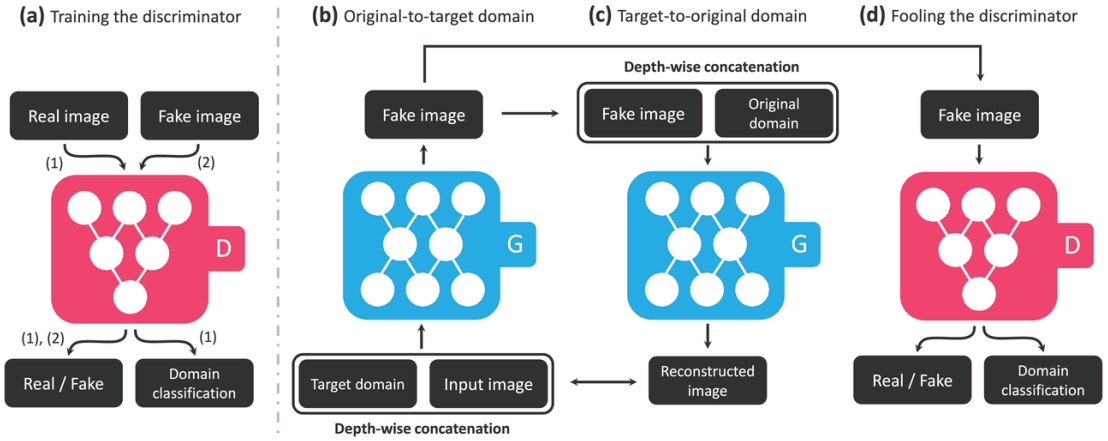


Figure 3.15.: The four terms of Equation 3.18 as training steps. (a) The discriminator learns to tell real from fake and to classify real images into their domains. (b) The generator takes the input image with the target domain label and produces a fake image. (c) The same generator maps that fake image back with the original label, which gives the reconstruction loss. (d) The generator learns to produce fakes the discriminator takes as real and classifies into the target domain. Adapted from (Choi et al. 2018, fig. 3).

3.8.3. StarGAN v2: diverse, domain-specific style codes

StarGAN v2 replaces the fixed label input to G with a style code injected through AdaIN. A mapping network $M_t(z)$ supplies latent-guided styles; a style encoder $E_t(x_{\text{ref}})$ supplies reference-guided styles. M, E, and D have domain-specific output heads, while G is shared (Choi et al. 2020, sec. 2; Ye 2026a, 123–25).

$$s_t = M_t(z) \text{ or } E_t(x_{\text{ref}}), \qquad \hat{x} = G(x, s_t), \qquad s_k = E_k(x). \tag{3.19}$$

Its key training terms, for $s_t^i = M_t(z_i)$, are

$$\begin{aligned} \mathcal{V}_{\text{adv}} &= \mathbb{E}_{x,k} \log D_k(x) + \mathbb{E}_{x,t,z} \log(1 - D_t(G(x, s_t))), \\ \mathcal{L}_{\text{sty}} &= \mathbb{E}\|s_t - E_t(G(x, s_t))\|_1, \\ \mathcal{L}_{\text{ds}} &= \mathbb{E}\|G(x, s_t^1) - G(x, s_t^2)\|_1, \\ \mathcal{L}_{\text{cyc}} &= \mathbb{E}\|x - G(G(x, s_t), s_k)\|_1, \\ \min_{G,M,E} \max_{D} \quad & \mathcal{V}_{\text{adv}} + \lambda_{\text{sty}}\mathcal{L}_{\text{sty}} - \lambda_{\text{ds}}\mathcal{L}_{\text{ds}} + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}}. \end{aligned} \tag{3.20}$$

Style reconstruction prevents ignoring the style input; the **minus** sign rewards different outputs; the cycle promotes source preservation. Unlike Diagonal GAN's thresholded

diversity loss, $\mathcal{L}_{\text{ds}}$ here is a distance reward. Recovering s_t from a generated image does not force two unrelated latent- and reference-derived styles to be equal.

Advantage: latent and reference control with multiple outputs per target domain. **Limitation:** multiple objectives must balance diversity and preservation, and trained domains still need labels. The original model has neither a text encoder nor text cross-attention; the transcript's diffusion/text comparison is an analogy.

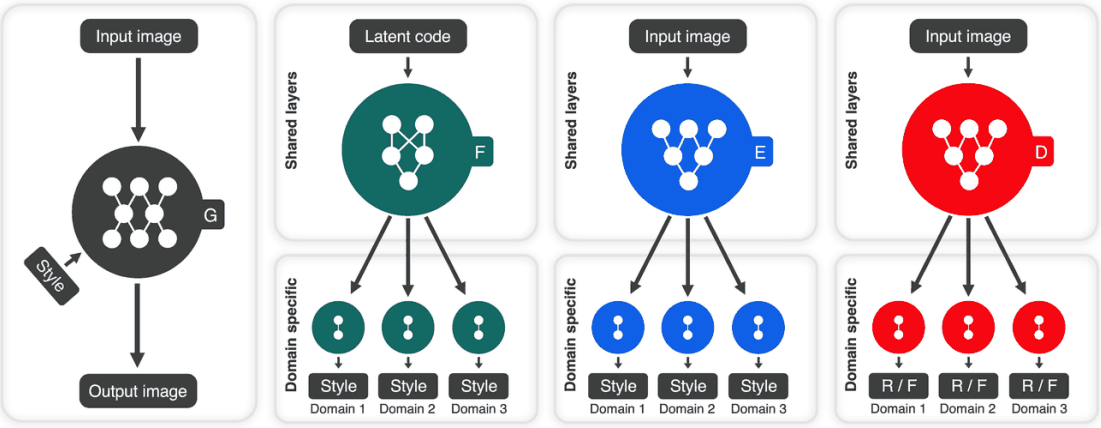


Figure 3.16.: The four modules of StarGAN v2. The generator translates an input image into an output reflecting a style code; the mapping network turns a latent code into style codes, one per domain; the style encoder extracts the style codes of a reference image; and the discriminator decides, for each domain, whether an image is a real image of that domain. All modules except the generator share their first layers and end in one branch per domain. Adapted from (Choi et al. 2020, fig. 2).

3.8.4. Comparison at a glance

Model	Input information	Added constraint or control	Main unresolved issue
Pix2Pix	Paired (x, y)	Paired L_1 reconstruction + conditional discriminator	Requires aligned pairs; limited multimodality
CycleGAN / DiscoGAN	Unpaired two-domain collections	Two adversarial directions + reconstruction cycle	Training stability and semantic ambiguity
RadialGAN	Heterogeneous domains	Shared latent + domain-specific encoders/decoders	Quality of the shared representation

Model	Input information	Added constraint or control	Main unresolved issue
StarGAN	One image + target domain label	Domain classification + shared-generator reconstruction	One output for fixed (x, t)
StarGAN v2	One image + latent/reference style	Style recovery + diversity + reconstruction	Diversity/preservation trade-off

The main change from StarGAN to StarGAN v2 is not simply "more domains." StarGAN says **which target domain** is desired; StarGAN v2 additionally supplies **which style inside that domain**. This makes latent-guided and reference-guided multimodal outputs possible while keeping a single shared image generator.

Training versus inference. Discriminator updates treat generated images as detached samples. Generator updates propagate gradients through the discriminator while holding its parameters fixed; the cycle also backpropagates through the participating generators. Inference uses the trained generator and its required conditioning modules, not the discriminator. The distinction between a minimax value and the loss minimized by each optimizer is therefore important.

3.9. Worked examples and reproducibility

3.9.1. AdaIN with actual numbers

Take one content channel $u = (1, 3)^\top$ and reference channel $v = (2, 6)^\top$. Their means are $2, 4$ and their population standard deviations are $1, 2$. Equation 3.3 gives

$$\mathrm{AdaIN}(u, v) = 2(u - 2\mathbf{1}) + 4\mathbf{1} = (2, 6)^\top.$$

The transformed channel has the target mean and variance. It does not follow that the output image equals the style image: only these feature statistics were constrained.

3.9.2. Matching marginals does not match correlations

Let $a = \sqrt{2}$ and choose centered two-channel feature matrices

$$X = \begin{bmatrix} a & 0 \\ -a & 0 \\ 0 & a \\ 0 & -a \end{bmatrix}, \qquad S = \begin{bmatrix} a & a/2 \\ -a & -a/2 \\ 0 & \sqrt{3/2} \\ 0 & -\sqrt{3/2} \end{bmatrix}.$$

Then $\Sigma_X = I_2$ and $\Sigma_S = \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix}$. Both channels already have mean zero and standard deviation one. AdaIN therefore leaves X unchanged and misses the target correlation. WCT multiplies X by $\Sigma_S^{1/2}$ and obtains the requested covariance. This is a finite-dimensional counterexample to interpreting AdaIN as full covariance matching.

The following self-contained NumPy example verifies these claims; it is a mathematical check, not a GAN training experiment.

```
import numpy as np

def cov(x):
    xc = x - x.mean(axis=0, keepdims=True)
    return xc.T @ xc / x.shape[0]

def spd_power(a, power):
    vals, vecs = np.linalg.eigh(a)
    if np.any(vals <= 0):
        raise ValueError("This example requires positive-definite covariance.")
    return (vecs * vals**power) @ vecs.T

a = np.sqrt(2.0)
x = np.array([[a, 0], [-a, 0], [0, a], [0, -a]])
s = np.array([[a, a/2], [-a, -a/2],
              [0, np.sqrt(1.5)], [0, -np.sqrt(1.5)]])
adain = (x - x.mean(0)) / x.std(0) * s.std(0) + s.mean(0)
wct = ((x - x.mean(0)) @ spd_power(cov(x), -0.5)
       @ spd_power(cov(s), 0.5) + s.mean(0))
assert np.allclose(cov(adain), np.eye(2))
assert np.allclose(cov(wct), cov(s))
print("AdaIN covariance:\n", np.round(cov(adain), 6))
print("WCT covariance:\n", np.round(cov(wct), 6))
```

3.10. Connections and further reading

The lecture's central connection is between **where a representation is transformed** and **what a translation objective constrains**. Normalization/AdaIN act mainly on the

channel axis, while spatial attention acts on the position axis; the unified $Y = AXT + B$ view makes this distinction explicit. StyleGAN, SNI, and Diagonal GAN then turn these transforms into architectural control signals at different resolutions. Pix2Pix, CycleGAN, StarGAN, and StarGAN v2 add progressively different kinds of supervision: paired targets, round-trip reconstruction, domain labels, and domain-specific style codes.

The main lecture should therefore be read as one progression rather than a list of GAN variants: **statistics -> axis-wise feature transforms -> controllable synthesis -> translation constraints -> scalable multi-domain conditioning**. The brief paper extensions above are useful related reading, but they are intentionally separated from the pp. 78–125/script narrative.

3.11. Advanced topics

The chapter above follows the lecture and its `Part 1 --- GAN` pages. This section develops the points the lecture named and left, and the derivations the main text states without giving. Each subsection is self-contained and none is needed to follow the main line.

3.11.1. Gram matrices, and why AdaIN is their diagonal case

Before AdaIN, neural style transfer represented style by correlations of fixed encoder features. With $\Phi_\ell(I) \in \mathbb{R}^{N_\ell \times C_\ell}$ and $\Gamma_\ell(I) = \Phi_\ell(I)^\top \Phi_\ell(I)$, a representative objective is (Gatys et al. 2016)

$$\mathcal{L}_{\text{style}} = \sum_\ell a_\ell \|\Gamma_\ell(I_o) - \Gamma_\ell(I_s)\|_F^2, \qquad \mathcal{L}_{\text{content}} = \|\Phi_{\ell_c}(I_o) - \Phi_{\ell_c}(I_c)\|_F^2. \tag{3.21}$$

A Gram matrix is unchanged by a permutation of spatial positions, so it captures channel co-occurrence statistics more readily than layout. **Advantage:** rich multi-layer appearance matching. **Limitation:** the original method optimizes each output image iteratively and does not guarantee content preservation. AdaIN is a feed-forward statistic-matching alternative, not an exact solver of the Gram objective.

That is the statement. Why a Gram matrix is a representation of style at all, and how it connects to the AdaIN the chapter has already used, is the part worth spelling out.

Write $\Phi_\ell(I) \in \mathbb{R}^{N_\ell \times C_\ell}$ for the features of a frozen VGG layer, rows indexed by position and columns by channel. Then $\Gamma_\ell = \Phi_\ell^\top \Phi_\ell$ has entries $\Gamma_{cc'} = \sum_n \Phi_{nc} \Phi_{nc'}$: an uncentred second-order statistic of the channels, summed over positions. Permuting the rows of Φ_ℓ leaves Γ_ℓ unchanged, which is the sense in which it discards layout; what survives is *which channels fire together*, and for a convolutional encoder a channel is a texture detector, so co-occurrence of channels is texture. That is the whole argument, and it also

bounds what the representation can do: anything that depends on where the texture is placed is invisible to it.

Which layer carries what is an empirical choice, and the usual one is that content is read off a single deep layer — `relu4_2` in Gatys et al. — where the representation is coarse enough to be insensitive to appearance, while style is read off several layers at once, `relu1_1` through `relu5_1`, because texture lives at every scale. The weights a_ℓ in Equation 3.21 balance them, and each layer's term is usually divided by $4N_\ell^2 C_\ell^2$ so that layers of different size contribute comparably.

The cost of the original method is the reason AdaIN exists. Gatys et al. produce each output by *optimising the image*: start from noise or from the content image and run L-BFGS on $\mathcal{L}_{\text{content}} + \mathcal{L}_{\text{style}}$ for hundreds of iterations, each one a forward and backward pass through VGG. Every new content-style pair pays this again.

Now the link. Take the diagonal of the Gram matrix and nothing else. Matching Γ_{cc} for every c matches each channel's second moment, and matching the feature means as well makes it each channel's mean and variance — which is exactly what Equation 3.3 does, in closed form, in one pass. AdaIN is therefore the diagonal special case of Gram matching, computed rather than optimised, and WCT of Equation 3.6 is the step that puts the off-diagonal entries back. The three sit on one axis: diagonal second moments (AdaIN), full covariance (WCT), full uncentred Gram across several layers (Gatys).

3.11.2. The feed-forward form of style transfer

Equation 3.3 is a transform, not a method: something has to turn the transformed features back into an image. Huang and Belongie train a decoder g against a frozen VGG encoder f, and the two losses are worth writing out because the first is not the obvious one.

With $t = \text{AdaIN}(f(c), f(s))$ the target features and $T = g(t)$ the output image,

$$\mathcal{L}_c = \|f(T) - t\|_2, \qquad \mathcal{L}_s = \sum_\ell \Big(\|\mu(\phi_\ell(T)) - \mu(\phi_\ell(s))\|_2 + \|\sigma(\phi_\ell(T)) - \sigma(\phi_\ell(s))\|_2 \Big), \tag{3.22}$$

minimising $\mathcal{L}_c + \lambda \mathcal{L}_s$ over the decoder alone. The content loss is the distance to the *AdaIN output* t, not to $f(c)$: the decoder is asked to reproduce the transformed features, not the original ones, which is what lets a single decoder serve every style. The style loss matches means and standard deviations across several VGG layers — the diagonal statistics of Section 3.11.1, again — rather than Gram matrices, and Huang and Belongie report the two give comparable results at lower cost.

Because the transform is explicit, two knobs come for free at test time and need no retraining. Interpolating t towards $f(c)$,

$$T_\alpha = g((1-\alpha)f(c) + \alpha\,\mathrm{AdaIN}(f(c), f(s))), \tag{3.23}$$

trades content against style continuously, with $\alpha = 0$ reconstructing the content image. Interpolating between several styles' AdaIN targets mixes styles. Neither is available to a decoder trained per style.

3.11.3. Why the W space disentangles better than Z

The chapter answers this in one sentence — the mapping can reshape the prior — and the standard account is worth the extra paragraph, because it explains *what goes wrong* in $\mathcal{Z}$ rather than only what is better in $\mathcal{W}$.

The prior on $\mathcal{Z}$ is fixed, $z \sim \mathcal{N}(0, I)$, and the generator must push it onto the data distribution. Suppose some combination of attributes is absent from the training data — long hair together with a masculine face, in the paper's example. The generator cannot map any region of $\mathcal{Z}$ to that combination, so the map from $\mathcal{Z}$ to attributes has to curve around the missing region. A curved map does not have axes: moving along a straight line in $\mathcal{Z}$ changes several attributes at once, which is entanglement, and it is forced by the shape of the data together with the fixed shape of the prior.

$\mathcal{W}$ is under no distributional constraint at all. Nothing requires $w = f(z)$ to be Gaussian, or to have any particular density, so the mapping network can absorb the warping: f takes the curvature and the synthesis network sees a space in which the factors can be closer to axis-aligned.

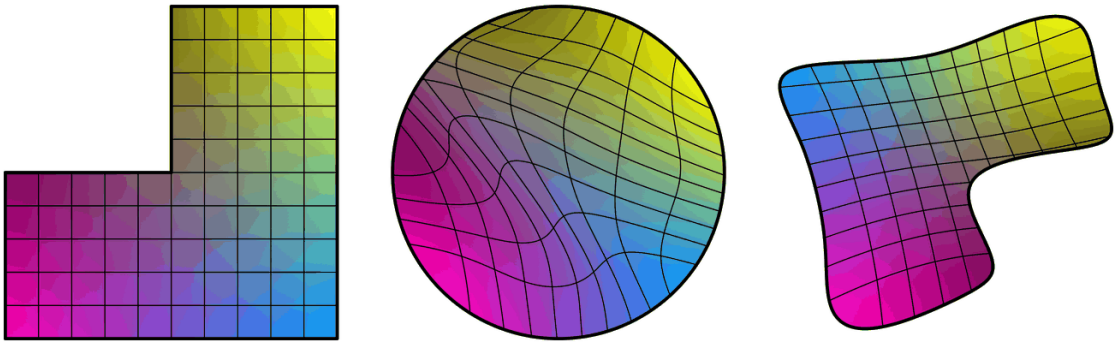


Figure 3.17.: The curvature argument, drawn for two factors such as masculinity and hair length. Left: a training set in which one combination is missing. Middle: the map from $\mathcal{Z}$ to features must curve around the missing region, so a straight line in $\mathcal{Z}$ changes several factors at once. Right: the learned map from $\mathcal{Z}$ to $\mathcal{W}$ undoes much of that warping. Adapted from (Karras et al. 2019, fig. 6).

The claim is that the warping has to happen somewhere and it is better done in a learned space than forced into the generator.

Two measurements support it, and both deserve their definitions rather than their names. **Perceptual path length** samples two latents, interpolates a short step ϵ between them, and measures the perceptual distance between the two images, averaged and normalised by ϵ^2:

$$\mathrm{PPL} = \mathbb{E}\left[\tfrac{1}{\epsilon^2}\, d_{\mathrm{VGG}}(G(\mathrm{lerp}(w_1, w_2; t)),\, G(\mathrm{lerp}(w_1, w_2; t+\epsilon)))\right]. \tag{3.24}$$

A small value says the image changes at a steady rate along the path — that the latent space is not curved — and it is the operational reading of "less entangled". **Linear separability** is the complementary test: train an auxiliary classifier on a binary attribute, then ask how well a *linear* hyperplane in latent space separates the two classes, reported as a conditional entropy. Both come out better for $\mathcal{W}$ than for $\mathcal{Z}$.

Two mechanisms in the paper follow directly. **Truncation** samples w and pulls it towards the mean, $w' = \bar{w} + \psi(w - \bar{w})$ with $\psi < 1$, trading variety for quality; it is done in $\mathcal{W}$ precisely because distances there are the perceptually meaningful ones. **Style mixing** draws two codes and uses one before a randomly chosen layer and the other after, during training, which stops the network from assuming that adjacent layers' styles are correlated — the regulariser that makes the per-resolution control of Equation 3.10 actually hold.

3.11.4. StyleGAN's noise, and what it is for

The chapter defines n_ℓ correctly and asserts its role. The evidence is an ablation, and it is worth stating because it is what separates *stochastic variation* from *style* as the chapter uses the words.

The attributes the noise controls are the ones a viewer cannot name as different: the exact placement of hair strands, freckles, skin pores, the precise outline of the background. Resampling the noise with w held fixed changes all of these and changes nothing a human would call the person's appearance. Removing the noise entirely does not give a cleaner image; it gives a flat, painterly one, because the generator has no source of variation for detail that ought to differ between samples and produces a featureless average instead.

The scale experiment is what makes it an argument about architecture rather than about texture. Noise fed only to the coarse layers moves large features — the overall curl of the hair; noise fed only to the fine layers moves only the finest detail. So the noise inherits the resolution of the layer it enters, exactly as the styles do in Equation 3.10, and the two channels are independent: w sets what the image is, n sets the arbitrary part of how it is realised.

3.11.5. StyleGAN2: demodulation and path-length regularization

StyleGAN2 removes explicit instance normalization from the synthesis path and modulates/demodulates convolution weights (Karras et al. 2020). If s_i is a style scale for input channel i,

$$w'_{ijk} = s_i w_{ijk}, \qquad w''_{ijk} = \frac{w'_{ijk}}{\sqrt{\sum_{i',k'} (w'_{i'jk'})^2 + \epsilon}}. \tag{3.25}$$

Demodulation controls the expected output scale without normalizing each observed feature map, addressing the characteristic droplet artifacts associated with the original normalization design. StyleGAN2 also uses path-length regularization,

$$\mathcal{L}_{\text{pl}} = \mathbb{E}_{w,r} \left(\| J_g(w)^\top r \|_2 - a \right)^2, \tag{3.26}$$

which encourages a more uniform sensitivity of the generated image to changes in w. **Advantage:** cleaner style modulation and fewer normalization artifacts. **Limitation:** these regularizers improve conditioning but do not guarantee semantic disentanglement.

Three points make the equations above an argument rather than a recipe.

Where the droplet comes from. Instance normalization standardises each feature map after the convolution, which means a generator can put a very large value in one spatial location and have the normalisation scale everything else down around it. The strong local signal survives, because it is what sets the scale; the rest of the map is suppressed. That is the droplet — a blob-shaped artefact that appears in the feature maps from 64×64 onwards and, once the network depends on it, in the output. Removing normalisation from the forward path removes the mechanism.

Why dividing by that particular norm works. Suppose the inputs to the convolution are independent with unit variance. After modulation the output j has variance $\sum_{i,k} (w'_{ijk})^2$, so dividing the weights by $\sqrt{\sum_{i',k'} (w'_{i'jk'})^2}$ restores unit output variance. The essential difference from instance normalization is the object being normalised: this acts on the **weights**, using the *expected* statistics under an assumption about the input, and not on the observed feature map. The generator therefore has no observed quantity to game, which is why the droplet does not come back.

The regulariser and its metric. In Equation 3.26 the constant a is not a hyperparameter but a running exponential average of the observed $\| J_g(w)^\top r \|_2$, so the term penalises deviation from the network's own current average sensitivity rather than from a chosen value. Its cost is a Jacobian-vector product, so it is applied *lazily*, once every 16 minibatches, with the step size scaled accordingly. And it is the same quantity the perceptual path length of Equation 3.24 measures: StyleGAN2 regularises the metric that StyleGAN introduced to diagnose entanglement, which is why the two move together.

StyleGAN2 also drops progressive growing, whose phase-in of new resolutions left a preference for fixed detail locations, in favour of skip and residual connections that train at full resolution throughout.

3.11.6. The solution set of unpaired translation

The chapter's two-point example shows that two bijections match the marginals. The situation is worse than "two", and saying how much worse is what justifies everything that follows it.

Let G satisfy $G_{\#}p_{\mathcal{X}} = p_y$ and let τ be any measure-preserving automorphism of $(\mathcal{Y}, p_y)$, that is, any map with $\tau_{\#}p_y = p_y$. Then

$$(\tau \circ G)_{\#}p_{\mathcal{X}} = \tau_{\#}(G_{\#}p_{\mathcal{X}}) = \tau_{\#}p_y = p_y, \tag{3.27}$$

so $\tau \circ G$ is just as valid. For a continuous p_y the group of such τ is infinite-dimensional, so the adversarial objective alone picks out nothing: its solution set is an orbit, not a point.

Cycle consistency shrinks the set but does not collapse it. Requiring $F \circ G \approx \mathrm{id}$ forces G to be approximately invertible, which removes the τ that destroy information; it does not remove the ones that are themselves invertible, and a swap of two equally likely modes is invertible. So what actually identifies the intended map is neither term. It is the architecture: a residual convolutional generator with a receptive field much smaller than the image, trained from an identity-like initialisation, finds small, local, low-frequency changes far more easily than a global rearrangement. The inductive bias is doing the identification, which is why the failures in Section 3.11.7 are the ones they are.

3.11.7. How cycle consistency fails

Two failures are named in the chapter. Both are worth developing, because each is a consequence of the objective rather than an accident of training.

Steganography. Chu et al. showed that a CycleGAN trained on aerial photographs and maps reconstructs features of the source that are not visible in the intermediate image at all: a building absent from the generated map reappears in the reconstruction. The generator has learned to encode the source in a low-amplitude, high-frequency signal that the map-domain discriminator cannot see and the reverse generator can read. The cycle loss is then near zero while the intermediate image is semantically wrong — the objective is satisfied and the task is not.

The mechanism follows from what is penalised. The cycle term constrains the *composition* $F \circ G$, not G alone, so any information channel that survives the round trip is rewarded, whether or not it is part of the visible translation. Adding noise to the intermediate image,

or otherwise destroying the high-frequency channel, degrades the reconstruction sharply — which is the test that distinguishes hidden information from honest translation.

Geometry. The original CycleGAN paper reports the limitation plainly: tasks requiring shape change fail while tasks requiring texture change succeed. Apple to orange keeps the apple's outline and repaints it; horse to zebra paints stripes on the horse, and on a rider as well if one is present, because nothing in the objective knows that a rider is not a horse; dog to cat produces minimal change. The reason is the same inductive bias as in Section 3.11.6, read the other way: a residual convolutional generator is well suited to recolouring a pixel from its neighbourhood and badly suited to moving content across the image, and a PatchGAN discriminator judges local patches, so it scores texture and is largely blind to global shape. Neither network is asked for geometry and neither provides it.

3.11.8. OT-driven CycleGAN and CycleGAN failure modes

A related optimal-transport formulation derives CycleGAN rather than modifying it (Sim et al. 2020). It is worth following the direction of that derivation, because it answers the question the lecture leaves open — where the cycle term comes from — and the answer is the opposite of what the shape of the objective suggests.

Start from a Kantorovich problem between the two domains, with a *joint* transport cost that charges a pair (x, y) for the round trip under the two maps $R : \mathcal{Y} \to \mathcal{X}$ and $H : \mathcal{X} \to \mathcal{Y}$,

$$d((x, y)) = \|x - R(y)\| + \|y - H(x)\|. \tag{3.28}$$

Minimising $\int d\, d\pi$ over couplings π of $p_{\mathcal{X}}$ and p_y is the primal problem, and its objective **is** the cycle/data-consistency term: Equation 3.28 is what $\mathcal{L}_{\text{cyc}}$ measures. The marginal constraints are then relaxed by Kantorovich duality, and it is the *dual* of those constraints that produces two Wasserstein-1 terms, each realised by a 1-Lipschitz critic. Collecting the two,

$$\min_{R,H} \lambda \mathcal{L}_{\text{cyc}}(R, H) + W_1(p_{\mathcal{X}}, R_{\#} p_y) + W_1(p_y, H_{\#} p_{\mathcal{X}}). \tag{3.29}$$

So the reading to carry away is not "W_1 matching, plus a cycle penalty of unexplained origin". It is: **the cycle term is the transport cost, and the adversarial terms are the penalties that enforce the marginals.** The weight λ is the exchange rate between the two, and taking λ large is what recovers the hard marginal constraints of the original Kantorovich problem. The critics are 1-Lipschitz because that is what the dual asks for, not because a Lipschitz constraint was found to stabilise training.

This remains a **surrogate training objective** once the ideal critic class is replaced by neural networks, so the equality between theory and implementation should not be overstated.

Cycle consistency also has important failure modes. First, it does not identify the intended semantic pairing: two domains with two equally likely elements admit a swapped bijection with perfect marginal matching and zero cycle error. Second, a network can hide source information in visually weak high-frequency signals and use it to reconstruct the input, producing a small cycle loss without meaningful visible preservation (Chu et al. 2017). Large geometric changes were also reported as difficult in the original CycleGAN study (Zhu et al. 2017). Thus realistic marginals plus a small cycle loss are not a certificate of correct correspondence.

3.11.9. StarGAN's mask vector

The chapter mentions the mask in a clause. It is what lets one generator train on two datasets whose label sets do not overlap — CelebA with facial attributes, RaFD with expressions — and the encoding is simple enough to give in full.

The label fed to the generator is a concatenation $\tilde{c} = [c_1, \dots, c_n, m]$, where c_i is the label vector for dataset i and m is a one-hot mask naming which dataset the current example came from. The entries of every c_i but the active one are set to zero and are to be ignored. The generator learns to read m and attend to the corresponding block, so a CelebA example carries expression bits that are zero and uninformative rather than zero and meaningful.

The discriminator's classification head is split to match: it predicts labels for both datasets, and the classification loss for an example is computed only over the head belonging to its own dataset. Without the mask the zeros would be indistinguishable from genuine negative labels, and the generator would learn that every CelebA face has a neutral expression.

3.11.10. CollaGAN: many inputs, one output

CollaGAN considers a different data regime from StarGAN: several **corresponding** domain observations of the same subject are available and one target domain is missing (Lee et al. 2019). With target domain t withheld,

$$\hat{x}_t = G(\{x_j : j \neq t\}, t), \qquad \mathcal{L}_{\text{mcyc}} = \mathbb{E}_t \sum_{j \neq t} \|x_j - \tilde{x}_{j|t}\|_1. \tag{3.30}$$

The multiple-cycle construction reconstructs each available input after inserting the predicted missing image. **Advantage:** it can fuse complementary observations from

several domains. **Limitation:** it requires corresponding multi-domain input sets, so it solves a different problem from single-input unpaired translation.

That is the summary. The setting deserves more, because it is the one where this lab's own work sits and because the objective is not a relabelling of StarGAN's.

The problem is imputation with several co-registered observations: N MR contrasts of the same slice, one of which is missing. A single-input translation model must pick one available contrast and discard the rest. CollaGAN takes all of them, so the generator is a map $G(\{x_i\}_{i\neq t}, t) \mapsto \hat{x}_t$, from every available input plus a target index to the missing one.

Cycle consistency has to be rebuilt for that signature, and the two-step form is the point of the paper. Having produced $\hat{x}_t$, put it back into the input set in place of the real x_t and ask for some *other* contrast j:

$$\tilde{x}_{j|t} = G(\{\hat{x}_t\} \cup \{x_i : i \neq t, j\},\ j), \qquad \mathcal{L}_{\text{mcc}} = \sum_{j\neq t} \left\| x_j - \tilde{x}_{j|t} \right\|_1. \tag{3.31}$$

A single round trip would only check that $\hat{x}_t$ is invertible back to where it came from; this *multiple* cycle checks that it is consistent with every other contrast at once, which is what a wrong but plausible $\hat{x}_t$ fails. The discriminator carries a classification head over contrasts, as in Equation 3.18, so the output must also be recognisable as contrast t. For MR synthesis the reconstruction term is supplemented by a structural-similarity term, since L_1 alone tolerates the loss of fine structure that matters clinically.

The reason the collaborative version wins is worth stating plainly: the contrasts are different measurements of the same anatomy, so what is missing from one is usually present in another. A model that reads only one of them throws away exactly the redundancy the problem provides.

Based on the scribe note of Jeongchan Kim, with material from Daegyu Sung, Seungwan Kang and Chaeyoung Oh; edited by Jong Chul Ye.

4. VAE, beta-VAE and Normalizing Flow

4.1. Overview

The previous chapters built generative models by *comparing* distributions: a discriminator, a divergence, a transport cost. This chapter builds them by *transforming* one. A latent variable is drawn from a simple prior and pushed through a decoder, and the whole difficulty moves into one question — how do you train a model whose likelihood you cannot write down?

Two answers follow from one construction. If the map from noise to latent code is cheap but not invertible in the data, you bound the likelihood and get the **variational autoencoder**. If instead you insist the map be invertible, the bound becomes an equality and you get a **normalizing flow**. The lecture derives both from a single encoder written as a deterministic function of noise, so VAE, beta-VAE, NICE, Real NVP and Glow are not five models but five choices of one object F_ϕ^x.

The exposition follows *Part 2 — VAE*, slides 3–41, and Chapter 13 of *Geometry of Deep Learning* (Ye 2026b, 2022). Equations carrying the book's numbering are marked where they first appear. A section of advanced topics at the end collects material the lecture did not develop, so the main narrative follows the class itself.

4.2. Notation

Symbol	Meaning
$x \in \mathcal{X} \subseteq \mathbb{R}^D$	An observation; D is the **data** dimension
$d\mu(x)$	The data measure; in training, the empirical measure of the dataset
$z \in \mathbb{R}^d$	A latent variable; d is the **latent** dimension
m	The **split index** of a coupling layer, $1 \le m < D$
θ, ϕ	Decoder and encoder parameters
$p(z)$	The latent prior, here $\mathcal{N}(0, I_d)$
$p_\theta(x \mid z)$	The conditional observation distribution

Symbol	Meaning
$G_\theta(z)$	The decoder mean, when the observation model is Gaussian
$q_\phi(z \mid x)$	The approximate posterior, or stochastic encoder
$u \sim r(u)$	Noise with density r; throughout, $u \sim \mathcal{N}(0, I_d)$
$F_\phi^x(u)$	The encoder: a map from noise to a latent sample, x fixed
$\mu_\phi(x), \sigma_\phi(x)$	The VAE encoder's mean and coordinatewise standard deviation
F_ϕ	The invertible data-to-latent map of a flow
$\sigma > 0$	The fixed scalar input-noise level in the flow construction
ℓ_{ELBO}	The negative ELBO, the quantity minimized

Three dimensions, three letters. The VAE compresses, so $d < D$. A bijective flow cannot, so there $D = d$; once that is said, flow sections write D and the letter d does not reappear. A coupling layer splits the D coordinates of its input into blocks of size m and $D - m$: the split index is m, never d. Keeping these apart matters because the same symbol is used for all three in much of the literature, including the lecture slides, which write a coupling split as $y_{1:d}, y_{d+1:D}$.

The data measure μ is not the encoder mean $\mu_\phi(x)$. In coordinate sums, $\mu_i(x)$ and $\sigma_i(x)$ drop the parameter subscript. The symbol $\odot$ is elementwise multiplication, I_m the $m \times m$ identity, and $\operatorname{diag}(a)$ the diagonal matrix built from a. Norms are Euclidean and all logarithms natural. For $f : \mathbb{R}^m \to \mathbb{R}^n$, $\partial f(v)/\partial v$ is the $n \times m$ Jacobian. Losses written with an argument x are per observation; objectives without one average over $d\mu(x)$.

4.3. The evidence lower bound

4.3.1. Why an approximate posterior is needed

The model draws $z \sim p(z)$ and then $x \sim p_\theta(x \mid z)$, so the joint density factors as $p_\theta(x, z) = p_\theta(x \mid z)p(z)$ and the quantity maximum likelihood needs is the marginal

$$p_\theta(x) = \int_{\mathbb{R}^d} p_\theta(x \mid z)p(z)\, dz. \tag{4.1}$$

Each factor is cheap. With the Gaussian decoder used throughout, $p_\theta(x \mid z) = \mathcal{N}(G_\theta(z), I_D)$, evaluating the integrand at a given z costs one decoder pass. The integral is the problem:

$$p_\theta(x) = \frac{1}{(2\pi)^{(D+d)/2}} \int_{\mathbb{R}^d} \exp\!\left(-\tfrac{1}{2}\|x - G_\theta(z)\|^2 - \tfrac{1}{2}\|z\|^2\right) dz.$$

The integrand is Gaussian in x but not in z, because G_θ is a nonlinear network and the exponent is therefore not quadratic in z. It has no closed form, and quadrature in a latent space of realistic dimension is out of reach. The exact posterior

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)p(z)}{p_\theta(x)}$$

is no help: its normalizer is the same integral. So we introduce a tractable family $q_\phi(z \mid x)$ and bound the likelihood instead of computing it.

4.3.2. Jensen's inequality

Fix x with $p_\theta(x) > 0$. The encoder $q_\phi(z \mid x)$ is chosen from a parameterized family; it is not obtained by evaluating the true posterior. Assume it is positive wherever $p_\theta(x, z)$ is, apart from null sets, so that sampling under q_ϕ omits no region contributing to Equation 4.1. A Gaussian encoder with positive variances satisfies this everywhere on $\mathbb{R}^d$.

Multiplying and dividing by $q_\phi(z \mid x)$ and applying concavity of the logarithm,

$$\begin{aligned} \log p_\theta(x) &= \log\left(\int q_\phi(z \mid x)\, \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}\, dz\right) \\ &\geq \int q_\phi(z \mid x) \log\left(\frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}\right) dz. \end{aligned} \tag{4.2}$$

Assuming the terms are integrable, the right-hand side splits into

$$\int \log p_\theta(x \mid z)\, q_\phi(z \mid x)\, dz - D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p(z)\big)\,.$$

This is the **evidence lower bound**. When $q_\phi(z \mid x) = p_\theta(z \mid x)$ the ratio inside the logarithm is the constant $p_\theta(x)$ and Jensen holds with equality; Section 4.12.1 identifies the gap in general.

4.3.3. Reconstruction against regularization

Negating gives the quantity to minimize:

$$\boxed{\begin{aligned}\ell_{\mathrm{ELBO}}(x,\theta,\phi) := &-\int \log p_\theta(x \mid z)\, q_\phi(z \mid x)\, dz \\ &+ D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p(z)\big)\,.\end{aligned}} \tag{4.3}$$

The first term is **reconstruction**: for a Gaussian decoder with identity covariance it is half the expected squared error between x and $G_\theta(z)$, up to a constant, and it measures error in observation space. The second is **regularization**: it compares the encoder with the prior in latent space, and it is what makes the model generative rather than merely a good autoencoder. Without it nothing ties the codes the encoder produces to the prior that generation samples from. KL divergence is nonnegative but asymmetric, so it is not a metric. Training minimizes $\int \ell_{\mathrm{ELBO}}\, d\mu(x)$.

4.4. The general encoder

Everything that follows is one objective specialized in two directions. The step that makes this possible is to stop treating the encoder as a density and start treating it as a *map applied to noise*.

4.4.1. Sampling written as a deterministic map

Let $u \sim r(u)$, in our case $\mathcal{N}(0, I_d)$. For fixed x the encoder turns u into a latent sample, and the density of that sample is

$$\boxed{q_\phi(z \mid x) = \int \delta\Big(z - F_\phi^x(u)\Big)\, r(u)\, du, \qquad u \sim r(u).} \tag{4.4}$$

This is equation (13.60) of the book. Sampling means drawing u and setting $z = F_\phi^x(u)$; inside an integral over z, the Dirac delta selects that value. The entire difference between a VAE and a flow will be the choice of F_ϕ^x — Figure 4.1 draws the Gaussian one.

4.4.2. The general loss

Fix x and ϕ, assume F_ϕ^x is a diffeomorphism and the logarithms below are integrable, and expand the KL term of Equation 4.3:

$$\begin{aligned}\ell_{\mathrm{ELBO}}(x,\theta,\phi) = &-\int \log(p_\theta(x \mid z)p(z))\, q_\phi(z \mid x)\, dz \\ &+\int \log q_\phi(z \mid x)\, q_\phi(z \mid x)\, dz.\end{aligned}$$

Both integrals are now rewritten through Equation 4.4.

The likelihood and prior term. Substituting the delta representation and integrating out z,

$$\begin{aligned}&-\int \log(p_\theta(x \mid z)p(z))\, q_\phi(z \mid x)\, dz \\ &\quad= -\iint \log(p_\theta(x \mid z)p(z))\, \delta\big(z - F_\phi^x(u)\big)\, r(u)\, du\, dz \\ &\quad= -\int \log p_\theta\big(x \mid F_\phi^x(u)\big)\, r(u)\, du - \int \log p\big(F_\phi^x(u)\big)\, r(u)\, du.\end{aligned}$$

The encoder log-density term. Here q_ϕ appears both inside and outside the logarithm, so give the inner copy its own variable u':

$$\begin{aligned}&\int \log q_\phi(z \mid x)\, q_\phi(z \mid x)\, dz \\ &\quad= \int \log\left(\int \delta\big(F_\phi^x(u) - F_\phi^x(u')\big)\, r(u')\, du'\right) r(u)\, du.\end{aligned}$$

The inner integral is evaluated by changing variables,

$$v = F_\phi^x(u'), \qquad u' = H_x(v), \qquad H_x = (F_\phi^x)^{-1},$$

where H_x inverts the noise-to-latent map at fixed x and ϕ. Differentiating $F_\phi^x(H_x(v)) = v$ gives the inverse-Jacobian relation, and the delta then selects $v = F_\phi^x(u)$, at which point $u' = u$:

$$\int \delta\big(F_\phi^x(u) - F_\phi^x(u')\big)\, r(u')\, du' = \frac{r(u)}{\left|\det\left(\dfrac{\partial F_\phi^x(u)}{\partial u}\right)\right|}. \tag{4.5}$$

Taking the logarithm splits this into an entropy term in r alone and a log-determinant term.

Collecting. The two integrals in $\log r(u)$ and $\log p(F_\phi^x(u))$ combine into a log ratio, leaving the book's equation (13.61):

$$\boxed{\begin{aligned}\ell_{\mathrm{ELBO}}(x,\theta,\phi) = &-\int \log p_\theta\big(x \mid F_\phi^x(u)\big)\, r(u)\, du \\ &+\int \log\left(\frac{r(u)}{p\big(F_\phi^x(u)\big)}\right) r(u)\, du \\ &-\int \log\left|\det\left(\frac{\partial F_\phi^x(u)}{\partial u}\right)\right| r(u)\, du.\end{aligned}} \tag{4.6}$$

The first integral is reconstruction; the last two together are the KL term. Nothing so far has assumed a Gaussian encoder, a particular prior, or any architecture. The three sections that follow each fix F_ϕ^x and read off the consequence.

4.5. The variational autoencoder

4.5.1. A Gaussian encoder

Take the encoder affine in the noise:

$$\boxed{z = F_\phi^x(u) = \mu_\phi(x) + \sigma_\phi(x) \odot u, \qquad u \sim \mathcal{N}(0, I_d).} \tag{4.7}$$

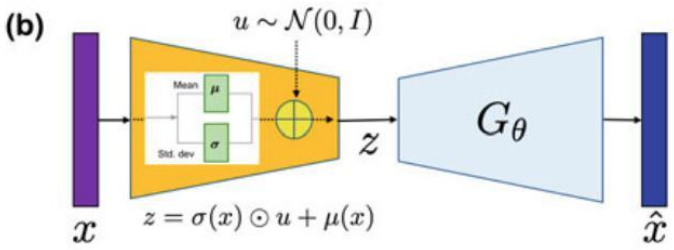


Figure 4.1.: The Gaussian VAE encoder. The network predicts $\mu_\phi(x)$ and $\sigma_\phi(x)$, the latent code is formed as $z = \sigma_\phi(x) \odot u + \mu_\phi(x)$ with $u \sim \mathcal{N}(0, I_d)$, and the decoder G_θ maps z back to the data space. From the course slides (Ye 2026b, 11).

Each coordinate of $\sigma_\phi(x)$ is positive, so Figure 4.1 realizes

$$q_\phi(z \mid x) = \mathcal{N}\big(\mu_\phi(x), \operatorname{diag}\left(\sigma^2_\phi(x)\right)\big)\,,$$

and the Jacobian the general loss asks for is immediate:

$$\frac{\partial F^x_\phi(u)}{\partial u} = \operatorname{diag}\left(\sigma_\phi(x)\right), \qquad \log\left|\det\left(\frac{\partial F^x_\phi(u)}{\partial u}\right)\right| = \frac{1}{2}\sum_{i=1}^{d}\log\sigma_i^2(x).$$

Note what is *not* required: the map from observations to codes need not be invertible. Only the map from noise to code, at fixed x, must be. This is the VAE of Kingma and Welling (Kingma and Welling 2014), and writing the sample this way is also what lets gradients reach ϕ — see Section 4.12.2.

4.5.2. The Gaussian KL term

Let r and p both be standard Gaussian. Their normalizing constants cancel in the ratio, so

$$\log\left(\frac{r(u)}{p(F^x_\phi(u))}\right) = \tfrac{1}{2}\left(\|\mu_\phi(x) + \sigma_\phi(x)\odot u\|^2 - \|u\|^2\right),$$

and since $\mathbb{E}[u_i] = 0$, $\mathbb{E}[u_i^2] = 1$, the middle term of Equation 4.6 averages to $\frac{1}{2}\sum_i(\mu_i^2(x) + \sigma_i^2(x) - 1)$. Adding the log-determinant gives the book's (13.64)–(13.65):

$$D_{\mathrm{KL}}\big(q_\phi(z \mid x)\,\|\,\mathcal{N}(0, I_d)\big) = \frac{1}{2}\sum_{i=1}^{d}\left(\sigma_i^2(x) + \mu_i^2(x) - \log\sigma_i^2(x) - 1\right). \tag{4.8}$$

The variance part, $\sigma_i^2 - \log\sigma_i^2 - 1$, is minimized at $\sigma_i^2 = 1$ and grows without bound in both directions, so the term penalizes a collapsed code and an over-dispersed one alike.

4.5.3. From likelihood to reconstruction error

With $p_\theta(x \mid z) = \mathcal{N}(G_\theta(z), I_D)$,

$$-\log p_\theta(x \mid z) = \tfrac{1}{2}\|x - G_\theta(z)\|^2 + \tfrac{D}{2}\log(2\pi).$$

Two things are being used here and both matter: the observation model is Gaussian, and its covariance is the identity. Neither is forced on us — a Bernoulli decoder on binarized images gives a cross-entropy instead, and the squared-error form would then

be wrong. Dropping the parameter-free constant and combining with Equation 4.8 gives the objective, the book's (13.67):

$$\boxed{\begin{aligned}\ell_{\mathrm{VAE}}(\theta,\phi) = \frac{1}{2}\int_{\mathcal{X}}\int \big\|x - G_\theta\big(\mu_\phi(x) + \sigma_\phi(x)\odot u\big)\big\|^2 \, r(u)\,du\,d\mu(x) \\ + \frac{1}{2}\sum_{i=1}^{d}\int_{\mathcal{X}} \left(\sigma_i^2(x) + \mu_i^2(x) - \log\sigma_i^2(x) - 1\right) d\mu(x).\end{aligned}} \tag{4.9}$$

4.5.4. Varying the decoder output

After training, resampling the noise at a fixed observation traces out a family of reconstructions, the book's (13.68):

$$\hat{x}(u) = G_\theta\big(\mu_\phi(x) + \sigma_\phi(x)\odot u\big), \qquad u \sim \mathcal{N}(0, I_d).$$

The encoder's mean and spread stay fixed while u moves the code. This is the concrete sense in which a VAE is generative where a deterministic autoencoder is not; to generate without any input, draw $z \sim p(z)$ instead.

4.6. The beta-VAE

Weighting the two terms differently gives the beta-VAE (Higgins et al. 2017), the book's (13.69):

$$\boxed{\begin{aligned}\ell_{\beta\text{-VAE}}(\theta,\phi) = \frac{1}{2}\int_{\mathcal{X}}\int \big\|x - G_\theta\big(\mu_\phi(x) + \sigma_\phi(x)\odot u\big)\big\|^2 \, r(u)\,du\,d\mu(x) \\ + \frac{\beta}{2}\sum_{i=1}^{d}\int_{\mathcal{X}} \left(\sigma_i^2(x) + \mu_i^2(x) - \log\sigma_i^2(x) - 1\right) d\mu(x).\end{aligned}} \tag{4.10}$$

Setting $\beta = 1$ recovers Equation 4.9. A larger β buys prior conformity with reconstruction detail. Section 4.12.4 derives β as a Lagrange multiplier on an information budget, which is where the choice stops looking arbitrary.

Figure 4.2 shows both halves of the trade at once. A coordinate is better aligned with a factor of variation when moving it moves that factor and little else, and the beta-VAE rows do this more consistently. They are also blurrier: under a squared-error objective a decoder that cannot distinguish fine detail averages it away. Stronger regularization *can* encourage interpretable structure; it does not guarantee it, and past some point it simply discards information.

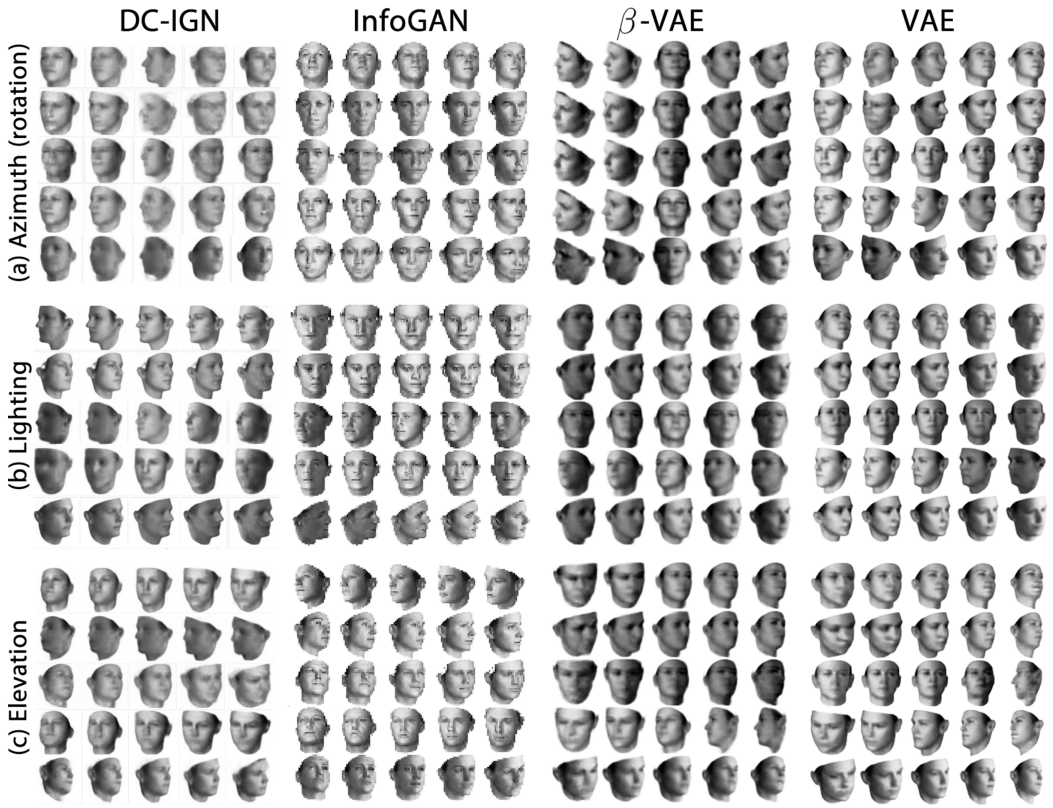


Figure 4.2.: Latent traversals on 3D faces. Each model occupies a group of five columns; within a row one latent coordinate is varied and the rest held fixed. The beta-VAE rows change one attribute at a time — azimuth, lighting, elevation — where the VAE rows change several together and InfoGAN separates them less cleanly. The beta-VAE faces are also visibly blurrier. From the course slides (Ye 2026b, 16), after Higgins et al. (Higgins et al. 2017).

4.7. Normalizing and invertible flows

4.7.1. Noise before an invertible map

Now make the encoder invertible and tie the decoder to it. Take $D = d$ and $F_\phi : \mathbb{R}^D \to \mathbb{R}^D$ invertible:

$$\boxed{F_\phi^x(u) = F_\phi(\sigma u + x), \qquad G_\theta = F_\phi^{-1}, \qquad u \sim \mathcal{N}(0, I_D).} \tag{4.11}$$

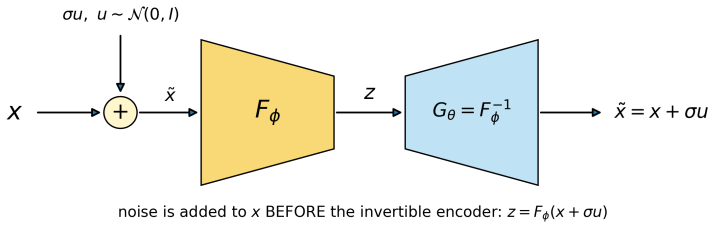


Figure 4.3.: A normalizing flow as an autoencoder. The noise σu is added to x *before* the invertible encoder F_ϕ, and the decoder is its inverse, so the round trip returns $\sigma u + x$. Redrawn after the course slides (Ye 2026b, 17), correcting panel (c), which draws the noise entering after the map.

Here $\sigma > 0$ is one fixed scalar, not the learned vector $\sigma_\phi(x)$ of the VAE, and the encoder and decoder now share parameters. As Figure 4.3 records, the noise enters *before* the map; for a nonlinear F_ϕ that is not the same as perturbing the output.

4.7.2. Why the reconstruction term becomes a constant

Because the decoder inverts the encoder exactly, reconstruction stops depending on the parameters at all:

$$\begin{aligned} \frac{1}{2}\int \left\| x - G_\theta\big(F_\phi^x(u)\big)\right\|^2 r(u)\, du &= \frac{1}{2}\int \left\| x - F_\phi^{-1}\big(F_\phi(\sigma u + x)\big)\right\|^2 r(u)\, du \\ &= \frac{1}{2}\int \|\sigma u\|^2 r(u)\, du = \frac{D\sigma^2}{2}. \end{aligned} \tag{4.12}$$

> **i** A constant worth getting right
>
> Since $u \sim \mathcal{N}(0, I_D)$ we have $\mathbb{E}\|u\|^2 = D$, so the value is $D\sigma^2/2$ — that is $\sigma^2/2$ *per coordinate*, which is how the slides write it (Ye 2026b, 18). The conclusion is the same either way, because what matters is only that the term does not depend on ϕ.

The entropy $\int \log r \, r \, du$ is likewise parameter-free. Dropping both leaves the flow loss,

$$\begin{aligned} \ell_{\text{flow}}(x, \phi) = & -\int \log p\Big(F_\phi^x(u)\Big)\, r(u)\, du \\ & -\int \log \left|\det\left(\frac{\partial F_\phi^x(u)}{\partial u}\right)\right| r(u)\, du, \end{aligned} \tag{4.13}$$

and with a standard Gaussian prior, up to one more constant,

$$\ell_{\text{flow}}(x, \phi) = \frac{1}{2}\int \|F_\phi(\sigma u + x)\|^2 r(u)\, du - \int \log \left|\det\left(\frac{\partial F_\phi(\sigma u + x)}{\partial u}\right)\right| r(u)\, du.$$

Two forces are now in balance and neither alone would do. The prior term pulls latent outputs into the high-density region; left to itself it would collapse everything onto the origin. The log-determinant is what stops it: a map that crushes volume pays for it. Generation draws $z \sim p(z)$ and applies $G_\theta = F_\phi^{-1}$.

4.7.3. Making the log-determinant tractable

A forward pass gives $F_\phi(x)$ cheaply, but the determinant of a general network's Jacobian does not come cheaply. Build F_ϕ from simple invertible layers instead:

$$F_\phi = f_K \circ \cdots \circ f_1, \qquad h_0 = x, \quad h_i = f_i(h_{i-1}), \quad h_K = F_\phi(x),$$

so that by the chain rule and multiplicativity of the determinant, the book's (13.73)–(13.75),

$$\boxed{\log \left|\det\left(\frac{\partial F_\phi(x)}{\partial x}\right)\right| = \sum_{i=1}^{K} \log \left|\det\left(\frac{\partial h_i}{\partial h_{i-1}}\right)\right|.} \tag{4.14}$$

Composition turns a product into a sum, as Figure 4.4 illustrates, but it does not by itself make any single term cheap. Each layer still needs a Jacobian whose structure gives up its determinant, while the composition stays expressive enough to be worth having. For the noisy-input objective the layers start at $h_0 = \sigma u + x$, and

$$\frac{\partial F_\phi(\sigma u + x)}{\partial u} = \left(\frac{\partial F_\phi(\sigma u + x)}{\partial x}\right) \sigma I_D,$$

so the perturbation contributes a constant $D \log \sigma$ and nothing else. Equation 4.13 is therefore the ordinary flow negative log density averaged over perturbed observations, up

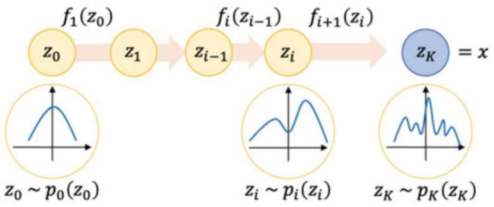


Figure 4.4.: A flow as a sequence of invertible maps, each reshaping the density of the previous one, from a simple p_0 to the data distribution p_K. The diagram runs in the generative direction $z_0 \to z_K = x$; density evaluation runs the other way. From the course slides (Ye 2026b, 19).

to parameter-free constants; Section 4.12.6 writes that density out and prices the dense determinant.

Generation reverses the order and inverts each layer, $F_\phi^{-1} = f_1^{-1} \circ \cdots \circ f_K^{-1}$, with f_K^{-1} acting first. The next two sections give layers for which both the inverse and the determinant are available in closed form.

4.8. NICE: additive coupling

4.8.1. The idea

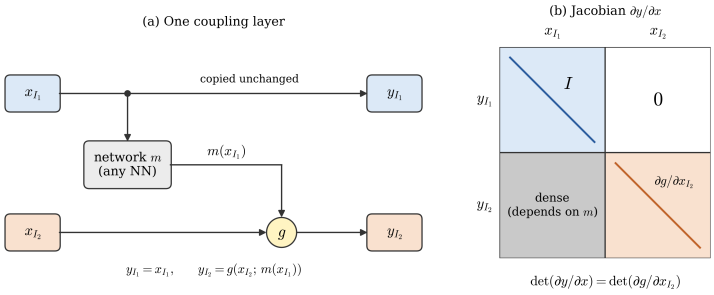


Figure 4.5.: The coupling construction. The first block passes through untouched; a network m reads it and sets the parameters of a simple invertible map g applied to the second block. Inverting needs g^{-1} only — m is never inverted, which is why it can be an arbitrary network. From the course slides (Ye 2026b, 30–31).

Figure 4.5 is the whole trick, and it is worth stating before any algebra. Split the coordinates. Copy one block. Let a network read the copied block and transform the other block with it. Because the copied block is available unchanged on both sides, the

network's output can be recomputed during inversion rather than inverted — so the network may be arbitrarily complicated while the layer stays exactly invertible.

4.8.2. Forward map, inverse, and Jacobian

Split x into $x_1 \in \mathbb{R}^m$ and $x_2 \in \mathbb{R}^{D-m}$ with $1 \le m < D$. NICE adds a network output to the second block (Dinh et al. 2015):

$$\boxed{y_1 = x_1, \qquad y_2 = x_2 + \mathcal{F}(x_1).} \tag{4.15}$$

The inverse recovers x_1 immediately and subtracts the same quantity,

$$\boxed{x_1 = y_1, \qquad x_2 = y_2 - \mathcal{F}(y_1),} \tag{4.16}$$

and the Jacobian is block triangular with identity diagonal blocks:

$$\frac{\partial y}{\partial x} = \begin{bmatrix} I_m & 0 \\ \dfrac{\partial \mathcal{F}(x_1)}{\partial x_1} & I_{D-m} \end{bmatrix}, \qquad \log \left| \det \left(\frac{\partial y}{\partial x} \right) \right| = 0. \tag{4.17}$$

4.8.3. The loss, and the price of volume preservation

A composition of additive couplings and permutations has $|\det(\partial F_\phi / \partial x)| = 1$, so every log-determinant in Equation 4.13 vanishes and, dropping constants,

$$\boxed{\ell_{\text{NICE}}(x, \phi) = \frac{1}{2} \int \|F_\phi(\sigma u + x)\|^2 r(u)\, du.} \tag{4.18}$$

Training simply drives perturbed observations into the prior's high-density region. The restriction is severe and easy to miss: additive layers can bend and reorient a region but can never change its volume, and stacking them does not help, since a product of ones is one. A flow built only from these layers cannot adapt the density of the prior to the density of the data, however complex the networks $\mathcal{F}$ are. The original NICE model therefore ends with a learned diagonal scaling layer, which supplies the missing volume change and its own log-determinant (Dinh et al. 2015). Real NVP puts the scaling inside every layer instead.

4.9. Real NVP: affine coupling

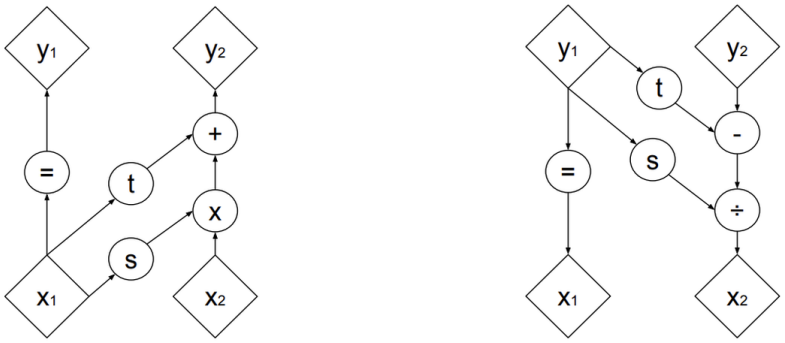
4.9.1. Forward map and inverse

Real NVP gives the coupling an input-dependent scale (Dinh et al. 2017):

$$\boxed{y_1 = x_1, \qquad y_2 = x_2 \odot \exp(s(x_1)) + t(x_1),} \tag{4.19}$$

with $s, t : \mathbb{R}^m \to \mathbb{R}^{D-m}$ producing log scales and translations. The exponential makes every scale positive, and $s = 0$ recovers NICE. Inversion evaluates the same two networks at $y_1 = x_1$:

$$\boxed{x_1 = y_1, \qquad x_2 = (y_2 - t(y_1)) \odot \exp(-s(y_1))\,.} \tag{4.20}$$



(a) Forward propagation (b) Inverse propagation

Figure 4.6.: Forward (a) and inverse (b) pass of an affine coupling layer. The unchanged branch supplies the same conditioning value in both directions, so s and t are evaluated, never inverted. From the course slides (Ye 2026b, 22, 33).

4.9.2. The determinant

Differentiating Equation 4.19, the lower-left block is whatever it is and the diagonal is what counts:

$$\frac{\partial y}{\partial x} = \begin{bmatrix} I_m & 0 \\ \dfrac{\partial y_2}{\partial x_1} & \operatorname{diag}(\exp(s(x_1))) \end{bmatrix}, \qquad \log\left|\det\left(\frac{\partial y}{\partial x}\right)\right| = \sum_j s_j(x_1). \tag{4.21}$$

A positive sum means the layer expands volume locally and a negative sum means it contracts. Section 4.12.8 carries a single layer through with numbers and checks this against a finite-difference Jacobian.

4.9.3. The loss

Let $F_\phi = f_K \circ \cdots \circ f_1$ be affine couplings with permutations between them, and write $h_0(u) = \sigma u + x$, $h_i(u) = f_i(h_{i-1}(u))$. Let $s_{i,j}$ be the jth log-scale output of layer i, evaluated on that layer's conditioning block $h_{i-1,1}(u)$. Permutations contribute nothing, so combining Equation 4.21 with Equation 4.14 and dropping constants,

$$\boxed{\ell_{\text{RealNVP}}(x, \phi) = \int \left[\frac{1}{2} \|F_\phi(\sigma u + x)\|^2 - \sum_{i=1}^{K} \sum_j s_{i,j}\big(h_{i-1,1}(u)\big) \right] r(u)\, du.} \tag{4.22}$$

Setting every log-scale network to zero returns Equation 4.18. The translation networks never appear in the determinant, yet they shape the loss through the states h_i. For ordinary density estimation, run the same layers at $h_0 = x$:

$$-\log p_X(x) = \frac{1}{2} \|F_\phi(x)\|^2 - \sum_{i=1}^{K} \sum_j s_{i,j}(h_{i-1,1}) + \frac{D}{2} \log(2\pi). \tag{4.23}$$

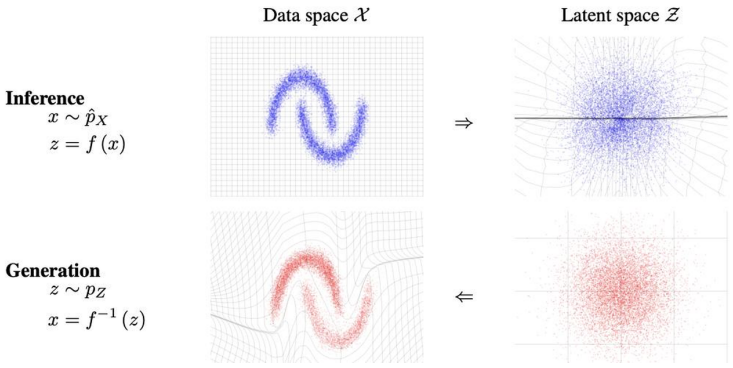


Figure 4.7.: Real NVP on two moons. Inference (top) carries the data to an approximately Gaussian cloud; generation (bottom) carries Gaussian samples back. The two directions are the same layers read forwards and backwards.

Figure 4.7 makes the point that a trained flow is one bijection: inference and generation are not two models but two ways of reading the same one.

4.10. Glow

The lecture closes with Glow (Kingma and Dhariwal 2018), which keeps the coupling layer and fixes what coupling alone cannot do. A coupling layer always leaves one block untouched, so some coordinates must be permuted between layers for information to reach everything. A fixed permutation is a crude instrument. Glow replaces it with a *learned* one and adds a normalization layer, giving three components per step (Ye 2026b, 23–25, 36–41).

Actnorm. A per-channel affine transform $y_{i,j} = s \odot x_{i,j} + b$ applied at every spatial position (i, j) of an $h \times w \times c$ tensor, initialized so that the post-transform activations have zero mean and unit variance on the first minibatch. Its log-determinant is $h \cdot w \cdot \mathrm{sum}(\log |s|)$.

Invertible 1×1 convolution. The learned replacement for a fixed permutation: $y_{i,j} = W x_{i,j}$ with W a $c \times c$ matrix shared across positions. Because the same matrix acts at every one of the $h \cdot w$ positions,

$$\log \left| \det \left(\frac{\partial \,\mathrm{conv2D}(h; W)}{\partial h} \right) \right| = h \cdot w \cdot \log |\det W| \,. \tag{4.24}$$

A permutation matrix is a special case, so the layer can only improve on one. Computing $\det W$ directly costs $O(c^3)$; parameterizing W in LU form removes that:

$$W = PL\,(U + \mathrm{diag}(s))\,, \qquad \log |\det W| = \mathrm{sum}\,(\log |s|)\,, \tag{4.25}$$

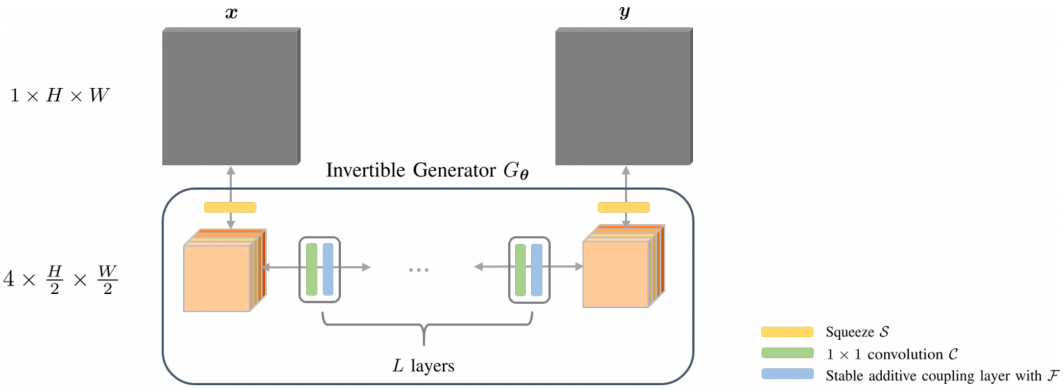
with P a fixed permutation, L unit lower triangular and U strictly upper triangular, bringing the cost to $O(c)$.

Affine coupling. Exactly Equation 4.19, acting on a channel split, with log-determinant $\mathrm{sum}(\log |s|)$.

Figure 4.8 and Figure 4.9 are worth reading together: the first says the model is a stack, the second says every element of the stack is a map whose inverse is written next to it. That is the whole design constraint of this chapter made concrete.

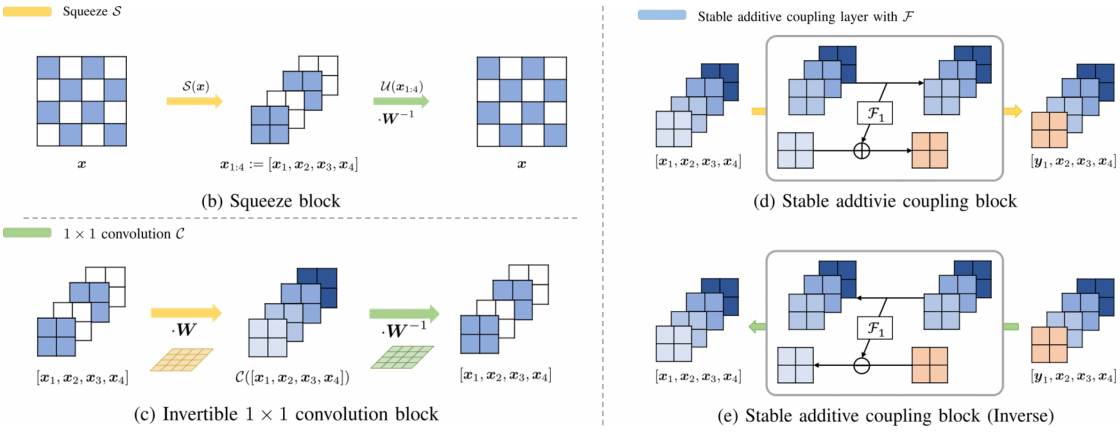
A **squeeze** operation between scales trades spatial resolution for channels, reshaping $1 \times H \times W$ into $4 \times \frac{H}{2} \times \frac{W}{2}$, so that the 1×1 convolution has channels to mix and the coupling split is spatially local. Stacking L such steps gives the invertible generator; every component has a closed-form inverse and an $O(c)$ log-determinant, so Equation 4.14 applies unchanged and the model is trained by exact maximum likelihood.

The point is not the particular three layers. It is that the whole chapter's constraint — invertible, with a cheap determinant — has enough room inside it to build a model that samples high-resolution images, provided each layer's structure is chosen so that its determinant is read off rather than computed.



(a) Overall architecture of Invertible Generator G_θ

Figure 4.8.: The invertible generator. An image is squeezed into channels, passed through L steps of 1×1 convolution and stable additive coupling, and squeezed back; every arrow runs in both directions. From the course slides (Ye 2026b, 38).



(b) Squeeze block

(c) Invertible 1×1 convolution block

(d) Stable addtivie coupling block

(e) Stable additive coupling block (Inverse)

Figure 4.9.: The three blocks, each with its inverse. (b) Squeeze rearranges a $H \times W$ map into four channels at half the resolution. (c) The invertible 1×1 convolution multiplies every spatial position by the same W, and is undone by W^{-1}. (d, e) The stable additive coupling layer adds $\mathcal{F}_1$ of the untouched block to the other, and subtracts it to invert. From the course slides (Ye 2026b, 39).

4.11. What the coupling construction costs

Flows must be expressive, invertible, and cheap to differentiate, and the three demands pull against each other. Setting $y_1 = x_1$ buys the triangular Jacobian, and the price is that no single layer can transform all coordinates jointly: each one leaves a block alone and moves the rest by a restricted affine map. Alternating the blocks lets every coordinate change eventually, but a restricted layer may need a deep stack to express what one unrestricted map would, and depth is compute. NICE pays more than Real NVP, being confined to unit volume.

Two limits are structural rather than architectural. A bijection forces $D = d$, so a flow has no bottleneck and no compressed representation — exactly what the VAE gives up invertibility to obtain. And scale factors that grow very large or very small make numerical inversion ill-conditioned, which bounds in practice how aggressive the learned volume changes can be.

4.12. Advanced topics

The material below was not developed in the lecture. It is collected here so that the preceding sections follow the class.

4.12.1. The ELBO gap, exactly

Write $\mathcal{L}(x, \theta, \phi) = -\ell_{\mathrm{ELBO}}(x, \theta, \phi)$. Bayes' rule gives a second derivation of the bound that also names the gap (Kingma and Welling 2014). Assuming finite expectations,

$$\begin{aligned} D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\big) &= \int q_\phi(z \mid x) \log\left(\frac{q_\phi(z \mid x) p_\theta(x)}{p_\theta(x, z)}\right) dz \\ &= \log p_\theta(x) - \mathcal{L}(x, \theta, \phi), \end{aligned}$$

so that

$$\log p_\theta(x) = \mathcal{L}(x, \theta, \phi) + D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\big)\,. \tag{4.26}$$

The KL term is nonnegative, which proves the bound, and it is *exactly* the gap, vanishing when the encoder matches the true posterior almost everywhere. Note which two distributions it compares: the encoder and the **posterior**. The regularizer in Equation 4.3 compares the encoder with the **prior**. They are different objects and only the first is the bound's slack.

4.12.2. Reparameterization and gradient estimation

Fix x and θ and write $f_\theta(z) = \log p_\theta(x \mid z)$. The reconstruction expectation depends on ϕ through the distribution it averages over, which is not something a sampler can differentiate. Reparameterization moves that dependence into the integrand:

$$\mathbb{E}_{q_\phi(z|x)}[f_\theta(z)] = \mathbb{E}_{u\sim r}\left[f_\theta\left(F_\phi^x(u)\right)\right],$$

and since r does not depend on ϕ, differentiating under the expectation gives the **pathwise** estimator

$$\nabla_\phi \mathbb{E}_q[f_\theta(z)] = \mathbb{E}_{u\sim r}\left[\left(\frac{\partial F_\phi^x(u)}{\partial \phi}\right)^\top \nabla_z f_\theta(z)|_{z=F_\phi^x(u)}\right]. \tag{4.27}$$

The alternative differentiates the density, using $\nabla_\phi q = q\nabla_\phi \log q$:

$$\nabla_\phi \int f_\theta(z) q_\phi(z \mid x)\, dz = \mathbb{E}_q\left[f_\theta(z)\nabla_\phi \log q_\phi(z \mid x)\right]. \tag{4.28}$$

This is the **score-function** estimator. Both are unbiased. The pathwise one uses $\nabla_z f_\theta$, first-order information about how the integrand moves when the code moves; the score-function one uses only the value $f_\theta(z)$, weighted by a score. For a smooth Gaussian problem the first is usually far less noisy, and Section 4.12.3 makes the gap concrete — but this is a tendency, not a theorem. The score-function estimator remains the one that works when no differentiable reparameterization exists, as with discrete latent variables, and a baseline independent of z reduces its variance at no bias, since $\mathbb{E}_q[\nabla_\phi \log q_\phi] = 0$.

4.12.3. Worked example: two gradient estimators

Take $q_\mu = \mathcal{N}(\mu, 1)$ and $f(z) = z$, so that $\nabla_\mu \mathbb{E}_{q_\mu}[z] = 1$ for every μ, and write $z = \mu + u$ with $u \sim \mathcal{N}(0, 1)$.

The pathwise estimator Equation 4.27 gives $\nabla_z f \cdot \partial z/\partial \mu = 1$ for *every* sample: exact, with zero variance. The score-function estimator Equation 4.28 uses $\nabla_\mu \log q_\mu(z) = z - \mu = u$, so one sample gives $f(z)u = (\mu + u)u = \mu u + u^2$, with mean 1 — also unbiased. For the variance, using $\mathbb{E}[u^2] = 1$, $\mathbb{E}[u^3] = 0$, $\mathbb{E}[u^4] = 3$:

$$\mathrm{Var}(\mu u + u^2) = \mu^2\mathbb{E}[u^2] + 2\mu\mathbb{E}[u^3] + \mathbb{E}[u^4] - 1 = \mu^2 + 2. \tag{4.29}$$

Table 4.1.: One-sample gradient estimates of $\nabla_\mu \mathbb{E}[z]$, $z \sim \mathcal{N}(\mu, 1)$, over 10^6 draws.

μ	score function: mean	score function: variance	$\mu^2 + 2$	pathwise: variance
0	1.0013	2.007	2	0.0
1	0.9988	3.003	3	0.0
2	0.9965	5.980	6	0.0
3	0.9972	10.999	11	0.0

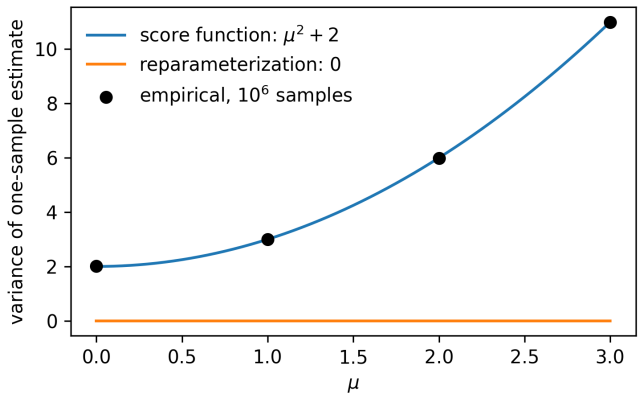


Figure 4.10.: Variance of one-sample estimates of $\nabla_\mu \mathbb{E}[z]$ for $z \sim \mathcal{N}(\mu, 1)$. Dots are the empirical values of Table 4.1, the curve is Equation 4.29, and the pathwise estimator sits on zero.

Table 4.1 and Figure 4.10 agree with Equation 4.29 to three figures. The variance grows with μ^2, and in a VAE the role of f is played by $\log p_\theta(x \mid z)$, a sum over thousands of pixels whose magnitude dwarfs μ here. That is the practical reason the VAE is trained with Equation 4.27.

4.12.4. The beta-VAE as constrained rate-distortion

Define average distortion and rate by

$$\mathcal{D}(\theta, \phi) = -\int \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)]\, d\mu(x), \qquad \mathcal{R}(\phi) = \int D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p(z)\big)\, d\mu(x).$$

Constraining the rate to a budget C and introducing a multiplier (Higgins et al. 2017),

$$\min_{\theta,\phi} \mathcal{D} \;\text{ s.t. }\; \mathcal{R} \le C \qquad \Longrightarrow \qquad \mathcal{J} = \mathcal{D} + \beta\,(\mathcal{R} - C)\,,\; \beta \ge 0,$$

and since $-\beta C$ is constant in the parameters, minimizing $\mathcal{J}$ is minimizing $\mathcal{D} + \beta\mathcal{R}$ — the beta-VAE objective (Burgess et al. 2018). Writing $\mathcal{D}^*(\mathcal{R})$ for the best distortion at a given rate, an interior smooth optimum satisfies $\beta = -d\mathcal{D}^*/d\mathcal{R}$, so β is the exchange rate between nats of latent information and reconstruction error. With nonconvex networks there is no guarantee that every budget C is attained by some fixed β.

The rate decomposes. With the aggregate encoder $q_\phi(z) = \int q_\phi(z \mid x)\, d\mu(x)$, splitting the log ratio gives

$$\mathcal{R} = I_q(X, Z) + D_{\mathrm{KL}}\big(q_\phi(z) \,\|\, p(z)\big)\,, \tag{4.30}$$

so the KL penalty is charging for two different things at once: the information the code keeps about the observation, and the mismatch between the aggregate code distribution and the prior.

4.12.5. Posterior collapse

Collapse is the state in which the latent variable carries nothing. At the extreme the decoder ignores z, so $p_\theta(x \mid z) = p_\theta(x)$, hence $p_\theta(z \mid x) = p(z)$; an encoder matching that posterior has $q_\phi(z \mid x) = p(z)$ and, by Equation 4.30, zero rate and zero mutual information. Partial collapse switches off some coordinates and not others.

The failure is most likely exactly where it is least wanted: when the decoder is powerful enough to model the data on its own, the KL term is a pure cost and the cheapest way to pay it is to stop using the code. Large β encourages this, but it happens at $\beta = 1$ too, and inference lag during joint training can drive a model there on its own (He et al. 2019). The symptoms under a standard Gaussian prior are $\mu_\phi(x) \approx 0$, $\sigma_\phi(x) \approx \mathbf{1}$ and a near-zero KL.

Figure 4.11 is a diagnosis, not a definition: spread along the diagonal says the encoder means still depend on the input. Means alone do not establish that two distributions agree, and blurry reconstructions on their own do not establish collapse. A common remedy is KL annealing, which starts training with a small weight on the KL term and raises it.

4.12.6. Exact flow densities and the cost of a determinant

For a differentiable bijection $z = F_\phi(x)$ with nonsingular Jacobian, let p_X be the density induced by pushing the prior through F_ϕ^{-1}; this is a model density, distinct from the data measure μ. Conservation of probability, $p_X(x)\, dx = p_Z(z)\, dz$, gives

$$-\log p_X(x) = -\log p_Z\big(F_\phi(x)\big) - \log \left| \det \left(\frac{\partial F_\phi(x)}{\partial x} \right) \right|. \tag{4.31}$$

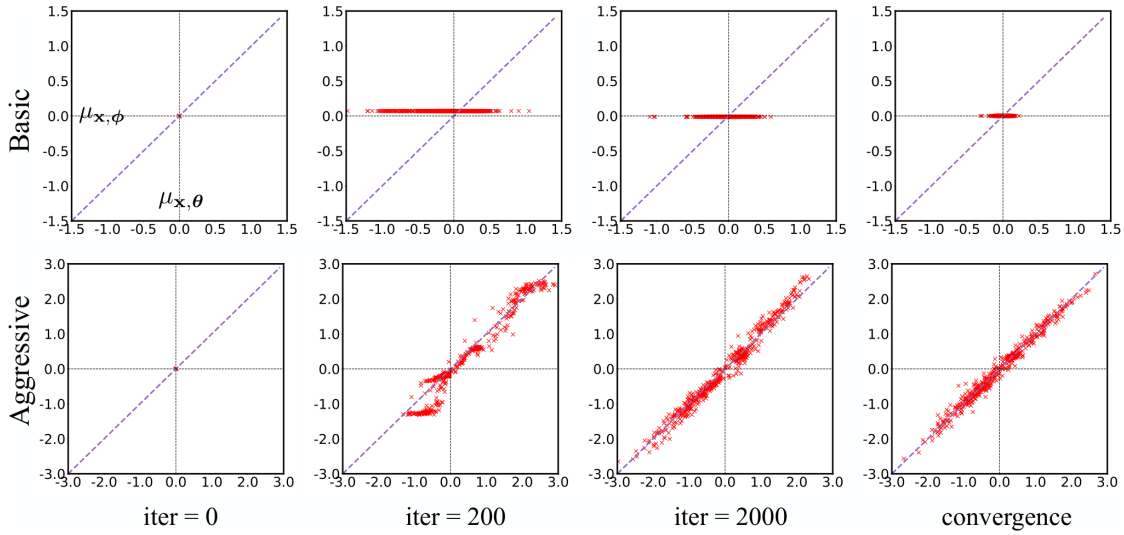


Figure 4.11.: Inference lag and posterior collapse. Posterior means for 500 validation observations, with the model-posterior mean on the horizontal axis and the encoder mean on the vertical; the diagonal marks agreement. Top: ordinary training — the model posterior spreads at iteration 200 while the encoder stays near zero, and by convergence both have returned to the origin. Bottom: extra encoder updates per decoder update let the encoder track the posterior, and the spread survives. After He et al. (He et al. 2019, fig. 2). The two rows use different axis ranges.

With a standard Gaussian prior the first term is $\frac{1}{2}\|F_\phi(x)\|^2$ plus a constant, recovering Equation 4.23. The noisy-input objective of Section 4.7 averages this at $x + \sigma u$ instead of at x.

The cost this chapter has been avoiding: a dense $D \times D$ Jacobian needs $O(D^2)$ storage and $O(D^3)$ operations for an LU-based determinant, before counting the differentiation that builds it. Decomposition makes the log-determinants additive, but additivity alone saves nothing unless each term is cheap (Rezende and Mohamed 2015).

4.12.7. Planar and autoregressive layers

Coupling is one way to meet the requirement; two others are standard.

Planar flows. A rank-one perturbation of the identity (Rezende and Mohamed 2015),

$$f(x) = x + v\,h(w^\top x + b)\,, \qquad v, w \in \mathbb{R}^D,\ b \in \mathbb{R},$$

with h a differentiable scalar activation. The matrix determinant lemma $\det(I + ab^\top) = 1 + b^\top a$ reduces the determinant to

$$\log\left|\det\left(\frac{\partial f(x)}{\partial x}\right)\right| = \log\left|1 + h'(w^\top x + b)\,w^\top v\right|, \tag{4.32}$$

an $O(D)$ computation. Invertibility is *not* automatic and the condition is easy to omit: for $h = \tanh$ it requires $w^\top v > -1$, without which the determinant can vanish or change sign and Equation 4.31 no longer applies. Under that condition, with $a = w^\top x + b$, the scalar map $a \mapsto a + (w^\top v)\tanh a$ has positive derivative and is onto, so a and then $x = y - v\tanh a$ are determined — but only through a numerical solve, unlike the closed-form inverse of a coupling layer. Planar layers are therefore used to enrich a variational posterior rather than to model data.

Autoregressive flows. Here x_i and y_i are scalar coordinates, not the vector blocks of the coupling sections. With $x_{<i} = (x_1, \dots, x_{i-1})$,

$$y_i = x_i \exp(s_i(x_{<i})) + t_i(x_{<i}), \qquad i = 1, \dots, D,$$

the Jacobian is triangular and $\log|\det \partial f/\partial x| = \sum_i s_i(x_{<i})$. Masked networks compute every s_i, t_i in one parallel pass when the full input is known, while the other direction is sequential because $x_{<i}$ must be recovered first. MAF takes the parallel direction for density evaluation; IAF takes it for sampling (Papamakarios et al. 2017; Kingma et al. 2016). A coupling layer is the two-block special case: less expressive per layer, fast in both directions.

Planar layers exploit a rank-one update, coupling layers a block-triangular split, autoregressive layers a coordinate ordering, and Glow a matrix factorization. In every case the determinant is made cheap by structure, not by computation.

4.12.8. Worked example: one affine coupling layer by hand

Two dimensions, $D = 2$ with split index $m = 1$. Let the flow be a single affine coupling with $s(x_1) = \frac{1}{2}x_1$ and $t(x_1) = -x_1^2$, so

$$z_1 = x_1, \qquad z_2 = x_2 \exp\left(\tfrac{1}{2}x_1\right) - x_1^2, \qquad z \sim \mathcal{N}(0, I_2). \tag{4.33}$$

Forward. At $x = (1, 0.5)$: $s = 0.5$, $t = -1$, so $z_1 = 1$ and $z_2 = 0.5e^{0.5} - 1 = -0.17564$.

Log-determinant. By Equation 4.21 it is just $s(x_1) = 0.5$. Writing the full Jacobian shows what drops out:

$$\frac{\partial z}{\partial x} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2}x_2 e^{x_1/2} - 2x_1 & e^{x_1/2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1.58782 & 1.64872 \end{bmatrix}, \qquad \det = e^{0.5} = 1.64872.$$

The lower-left entry involves both s and t; the determinant never sees it.

Log-likelihood. By Equation 4.31 with $p_Z = \mathcal{N}(0, I_2)$,

$$\log p_X(x) = -\log 2\pi - \tfrac{1}{2}\|z\|^2 + s(x_1) = -1.83788 - \tfrac{1}{2}(1 + 0.03085) + 0.5 = -1.85330.$$

Inverse. $x_2 = (z_2 - t(z_1))e^{-s(z_1)} = (-0.17564 + 1)e^{-0.5} = 0.5$, recovering x exactly, with s and t evaluated and never inverted.

Table 4.2.: One affine coupling layer at $x = (1, 0.5)$, each value checked numerically.

Quantity	By hand	Numerical check
$z = F(x)$	$(1, -0.17564)$	$(1, -0.17564)$
$\log\lvert\det \partial z/\partial x\rvert$	$s(x_1) = 0.5$	$\log 1.64872 = 0.5$ (finite differences)
$\log p_X(x)$	-1.85330	-1.85330
$F^{-1}(F(x))$	$(1, 0.5)$	$(1, 0.5)$
$\int p_X\, dx$	1	0.9999999981

The normalization in Table 4.2 is computed one x_1 at a time, since $\int p_X(x_1, x_2)\, dx_2 = \mathcal{N}(x_1; 0, 1)$ for every x_1. Figure 4.12 shows what one layer does: z_2 is shifted by $-x_1^2$

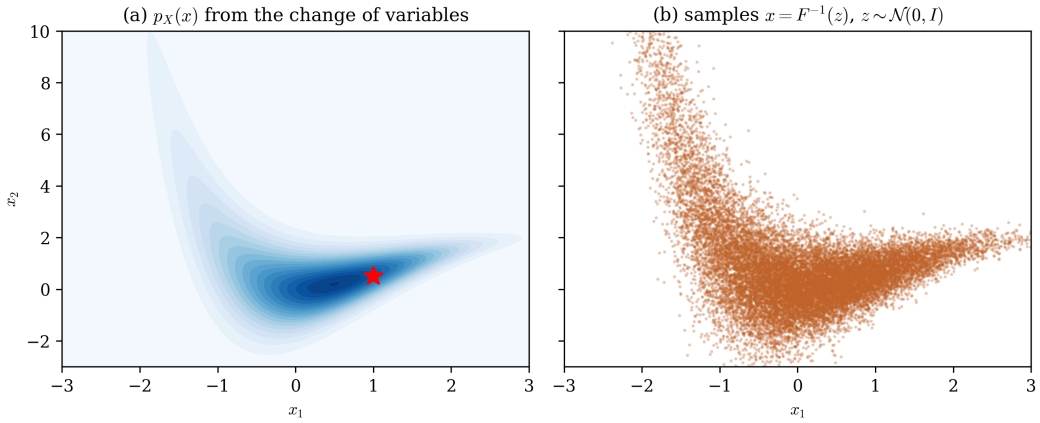


Figure 4.12.: One affine coupling layer applied to a standard Gaussian. (a) The density p_X from Equation 4.31; the star marks $x = (1, 0.5)$ of Table 4.2. (b) Samples $x = F^{-1}(z)$ with $z \sim \mathcal{N}(0, I)$. That the two agree is the change-of-variables formula at work.

and rescaled by $e^{x_1/2}$, so the Gaussian bends into a curved ridge. One layer can only bend x_2 as a function of x_1. Alternating the roles of the two coordinates across layers is what lets a flow bend in both directions.

4.12.9. Where this chapter leads

Table 4.3.: The models of this chapter as choices of the encoder in Equation 4.6.

Model	Encoder F^x_ϕ	dim z	Objective	log|det| cost	Inverse
VAE (Kingma and Welling 2014)	Gaussian, Equation 4.7	$d < D$	ELBO	closed form	not needed
beta-VAE (Higgins et al. 2017)	same	$d < D$	$\mathcal{D} + \beta\mathcal{R}$	closed form	not needed
NICE (Dinh et al. 2015)	additive coupling	D	exact likelihood	0	one pass
Real NVP (Dinh et al. 2017)	affine coupling	D	exact likelihood	$O(D)$	one pass

Model	Encoder F_ϕ^x	dim z	Objective	log|det| cost	Inverse
Glow (Kingma and Dhariwal 2018)	coupling + 1×1 conv	D	exact likelihood	$O(c)$ per conv	one pass
Planar (Rezende and Mohamed 2015)	Equation 4.32	D	posterior only	$O(D)$	numerical solve
MAF / IAF (Papamakarios et al. 2017; Kingma et al. 2016)	autoregressive	D	exact likelihood	$O(D)$	sequential one way

Read Table 4.3 down the dim z column and the chapter's one trade-off is visible: the VAE compresses and pays with an approximate likelihood; every flow keeps the likelihood exact and pays by giving up the bottleneck.

Diffusion models, the subject of the next part, take the flow's picture of generation as a chain of invertible steps and push it to the continuous limit. The chain becomes a stochastic differential equation, the sum of log-determinants becomes an integral, and the network defining the dynamics no longer needs the explicitly invertible coupling architecture that NICE, Real NVP and Glow were built around (Y. Song et al. 2021).

Based on the scribe note of Jungwoo Park, with material from Youngjin Shin and Daehan Yoon; edited by Jong Chul Ye.

Part II.

Diffusion Models

5. Self-supervised Denoising

5.1. Overview

The previous lectures built generative models that learn a data distribution directly: GANs through an adversarial game, VAEs through the ELBO, and normalizing flows through an exact change of variables. This lecture takes a step sideways to a problem that looks much more modest, image denoising, and asks a question that turns out to lead straight into the diffusion half of the course: *can a network learn to remove noise when it has never seen a clean image?*

We first place denoising inside the general inverse problem and recall the classical answer (regularized least squares and BM3D) and the first deep answer (supervised training, DnCNN). Supervised training needs clean targets, which are often physically impossible to obtain. Three families of methods avoid them: Noise2Noise trains on pairs of independent noisy copies; Noise2Self and Noise2Void train on a single noisy copy with a blind-spot network; SURE trains on a single noisy copy with a divergence penalty. These look like unrelated heuristics. The second half of the lecture shows that they are not: Tweedie's formula says that the Bayes-optimal denoiser is determined by the *score* $\nabla_y \log p(y)$ of the noisy data. For Gaussian noise all three methods can then be understood through the relation between the score and the conditional mean that each of them targets. Noise2Score makes this explicit, estimates the score once, and then applies the Tweedie formula that matches the noise model (Gaussian, Poisson, Gamma, and others).

The same decomposition — learn a score during training, use it through Tweedie's formula at inference — is the core of diffusion models, which is why this lecture sits between the VAE/flow lectures and DDPM. After reading this chapter you should be able to prove why Noise2Noise and Noise2Self do not collapse, derive SURE from Stein's lemma, derive Tweedie's formula for any exponential-family noise model, and explain why "denoising is score estimation".

5.2. Notation

We follow the notation of the slides and of *Geometry of Deep Learning* (Ye 2022).

Symbol	Meaning
$x \in \mathbb{R}^n$	the unknown clean image, flattened to n pixels
$y \in \mathbb{R}^n$	the noisy measurement
w	additive noise; in the Gaussian case $w \sim \mathcal{N}(0, \sigma^2 I)$
$\mathcal{A}$	forward (imaging) operator of a general inverse problem
F_Θ	denoising network with parameters Θ; $\hat{x} = F_{\Theta^*}(y)$
$\mathcal{R}(y;\Theta)$, R_Θ	network in residual form: the DnCNN noise predictor, or a score network
P_Y	distribution of the noisy data; $p(y)$ its density
$l(y) = \log p(y)$	log-density of the noisy data
$l'(y) = \nabla_y \log p(y)$	the **score function** of the noisy data
$\eta, T(y), p_0(y), \varphi(\eta)$	natural parameter, sufficient statistic, base measure and log-partition function of an exponential family
$l_0(y) = \log p_0(y)$	log of the base measure; $l_0'(y)$ its derivative
$\lambda(y) = \log \frac{p(y)}{p_0(y)}$	log-ratio appearing in the posterior of η
$\operatorname{div}_y F$	divergence $\sum_i \partial F_i / \partial y_i$, the trace of the Jacobian
σ_a	the *artificial* noise level added in a denoising autoencoder

References of the form "slide 12" point to the 38-page lecture deck *Self-supervised Image Denoising* (Ye 2026c), so that each section can be checked against the lecture. Norms are Euclidean. The slides write the score both as $l'(y)$ and $\ell'(y)$; we use $l'(y)$ throughout. A prime on a function of a vector y means its gradient. Expectations without a subscript are over everything that is random in the expression.

5.3. Denoising as an inverse problem

5.3.1. The forward model and ill-posedness

An imaging system (slide 3) — a microscope, a telescope, a CT scanner — maps an unknown object x to a measurement through a forward operator $\mathcal{A}$, and the measurement is corrupted by noise (Figure 5.1):

$$y = \mathcal{A}(x) + w. \tag{5.1}$$

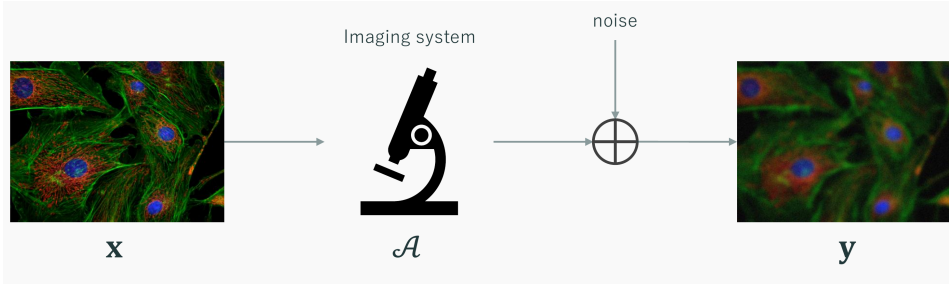


Figure 5.1.: An imaging system as a forward model. The clean object x passes through the imaging operator $\mathcal{A}$ (here a fluorescence microscope) and noise is added, giving the measurement y. Recovering x from y is the inverse problem. From slide 3 of the course slides (Ye 2026c), courtesy of the 2023 ISBI tutorial of Song, Chung and Ye.

In the fluorescence microscopy example of the lecture, $\mathcal{A}$ is convolution with the optical point-spread function, which blurs everything below the diffraction limit. Recovering x from y is an *inverse problem*, and it is typically *ill-posed*: many different x produce the same blurred and noisy y, so the data alone do not determine the answer. The question is not "what is the solution" but "which of the infinitely many candidates is the best one, and in what sense".

Denoising is the special case $\mathcal{A} = I$:

$$y = x + w. \tag{5.2}$$

It is ill-posed for the same reason — every x is consistent with y for some noise realization — and it is the building block for the general case, because most modern solvers for Equation 5.1 alternate a data-consistency step with a denoising step.

5.3.2. Classical regularization

The classical remedy (slide 4) restricts the solution space with prior knowledge:

$$\min_x \ \frac{1}{2}\|y - \mathcal{A}(x)\|_2^2 + \lambda R(x). \tag{5.3}$$

The first term is the *data fidelity*; the second is a *regularizer* that is small for plausible images. Common choices in the lecture were sparsity ($R(x) = \|x\|_1$, natural in fluorescence imaging where most pixels are dark), total variation ($R(x) = \sum_i \|(\nabla x)_i\|$, which favours

piecewise-constant images), and *regularization by denoising* (RED), where an off-the-shelf denoiser D defines $R(x) = \frac{1}{2}x^\top(x - D(x))$ (Romano et al. 2017).

It is worth seeing that Equation 5.3 is a Bayesian estimate in disguise, because the second half of this lecture is entirely Bayesian. With Gaussian noise $w \sim \mathcal{N}(0, \sigma^2 I)$ the likelihood is $p(y \mid x) \propto \exp(-\|y - \mathcal{A}(x)\|^2/2\sigma^2)$. If the prior is $p(x) \propto \exp(-\lambda R(x)/\sigma^2)$, Bayes' rule gives

$$-\log p(x \mid y) = \frac{1}{2\sigma^2}\|y - \mathcal{A}(x)\|^2 + \frac{\lambda}{\sigma^2}R(x) + \text{const},$$

so minimizing Equation 5.3 is the same as maximizing the posterior (MAP estimation). The regularizer is, up to the factor λ/σ^2 and an additive constant, a *negative* log-prior. Choosing R by hand is choosing the prior by hand; learning-based methods instead learn it from data. The score-based view at the end of this chapter is related but not identical: it learns the score $\nabla_y \log p(y)$ of the *noisy* data, which differs from the gradient of the clean log-prior $\log p(x)$ unless the noise vanishes.

5.3.3. BM3D: collaborative filtering of similar patches

For roughly a decade after its publication, BM3D (Dabov et al. 2007) (slides 5–7) was the method to beat in Gaussian denoising. Its idea is non-local: natural images repeat themselves, so a small patch usually has many look-alikes elsewhere in the image (Figure 5.2). BM3D works in two stages, each with three steps (Figure 5.3).

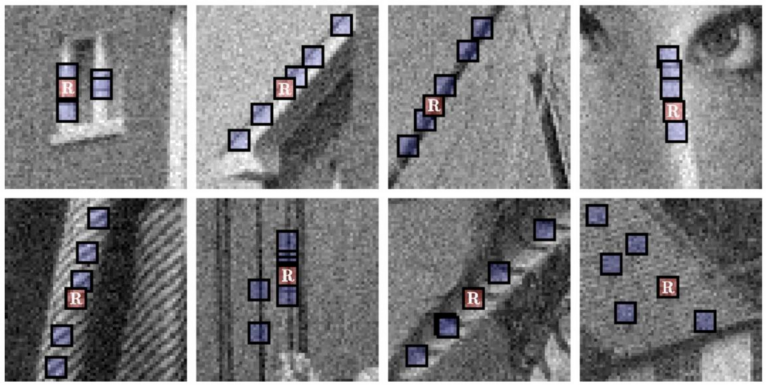


Figure 5.2.: Grouping by block matching on images corrupted by Gaussian noise ($\sigma = 15$). In each fragment, a reference block is marked "R" and the blocks matched to it are outlined; similar blocks are found along edges and in repeated textures, often far from the reference. From slide 5 of the course slides (Ye 2026c), originally Fig. 1 of Dabov et al. (2007).

1. **Grouping by block matching.** For a reference block R, search a window around it for blocks whose distance to R is below a threshold and stack them into a 3D array.
2. **Collaborative filtering.** Apply a 3D decorrelating transform to the stack (a 2D transform within each block and a 1D transform across the stack), shrink the coefficients, and invert the transform. In stage 1 the shrinkage is hard thresholding; in stage 2 it is Wiener filtering, using the stage-1 output as a pilot estimate of the signal spectrum.
3. **Aggregation.** Every pixel belongs to many groups, so it receives many estimates; these are averaged with weights that favour groups with less residual noise.

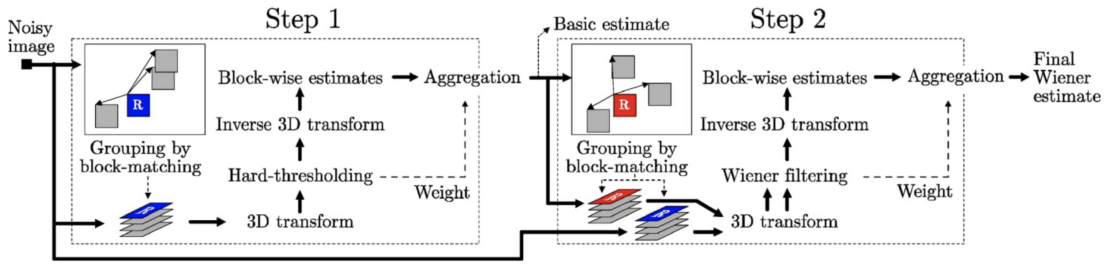


Figure 5.3.: The two stages of BM3D. Step 1 groups blocks, applies a 3D transform with hard thresholding and aggregates the block-wise estimates into a basic estimate. Step 2 repeats the grouping on the basic estimate and replaces hard thresholding by Wiener filtering, giving the final estimate. The dashed boxes are repeated for every reference block. From slide 6 of the course slides (Ye 2026c), originally Fig. 3 of Dabov et al. (2007).

The lecture gave the intuition through principal components: in a stack of similar blocks, the content shared across blocks lives in a few leading components, while white noise spreads evenly over all of them, so keeping the leading components and discarding the rest removes mostly noise. (The original paper uses fixed transforms such as the DCT and a Haar transform across the stack rather than PCA, but the reasoning is the same.) On standard test images at $\sigma = 25$ (input PSNR 20.18 dB), BM3D reaches 32.08 dB on *Lena* and 30.73 dB on *Barbara*.

The lecture also pointed out a modern reading of BM3D. To process a block, it finds other blocks that are similar to it and processes them together. That is what an attention layer does: a query is compared against keys, and the most similar values are combined. BM3D hand-codes with signal processing what a transformer learns.

5.4. Supervised denoising and why it is not enough

5.4.1. The supervised objective

Given pairs (x, y) of clean and noisy images, the most direct approach (slide 8) trains a feed-forward network to map y to x:

$$\min_{\Theta} \mathbb{E}_{y\sim P_Y}\, \ell_{sup}(\Theta), \qquad \ell_{sup}(\Theta) = d(x, F_\Theta(y)) = \|x - F_\Theta(y)\|^2, \tag{5.4}$$

and the trained network is used as $\hat{x} = F_{\Theta^*}(y)$. It is the simplest and fastest method — one forward pass per image — and, as we will use repeatedly, its population minimizer is known in closed form.

Lemma 5.1 (The squared loss is minimized by the conditional mean)**.** *Let (x, y) be jointly distributed with $\mathbb{E}\|x\|^2 < \infty$. Among all measurable functions F, the risk $\mathbb{E}\|x - F(y)\|^2$ is minimized by $F^*(y) = \mathbb{E}[x \mid y]$.*

Proof. Write $m(y) = \mathbb{E}[x \mid y]$ and expand around it:

$$\mathbb{E}\|x - F(y)\|^2 = \mathbb{E}\|x - m(y)\|^2 + \mathbb{E}\|m(y) - F(y)\|^2 + 2\,\mathbb{E}\big[(x - m(y))^\top (m(y) - F(y))\big].$$

Condition the cross term on y. Given y, the factor $m(y) - F(y)$ is fixed, and $\mathbb{E}[x - m(y) \mid y] = 0$ by definition of m, so the cross term is zero. The first term does not depend on F and the second is non-negative and vanishes exactly when $F = m$ (almost surely). □

So a perfectly trained supervised denoiser computes the **posterior mean**, or minimum mean squared error (MMSE) estimate. Keep this in mind: every self-supervised method below will be judged by whether its minimizer is still $\mathbb{E}[x \mid y]$.

5.4.2. DnCNN and residual learning

DnCNN (Zhang et al. 2017) (slides 9–10) made this work with a deep CNN. Its key design choice is *residual learning*: instead of predicting the clean image, the network $\mathcal{R}(y; \Theta)$ predicts the noise $y - x$, and the loss over N training pairs is

$$\ell(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \big\|\mathcal{R}(y_i; \Theta) - (y_i - x_i)\big\|_F^2, \tag{5.5}$$

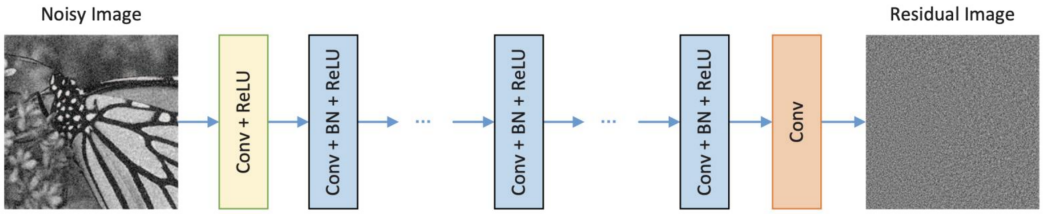


Figure 5.4.: The DnCNN architecture. The input is the noisy image and the output is the *residual*, i.e. the estimated noise, which is subtracted from the input to obtain the denoised image. From slide 9 of the course slides (Ye 2026c), originally Fig. 1 of Zhang et al. (2017).

with the denoised image $\hat{x} = y - \mathcal{R}(y; \Theta)$. The architecture is a plain stack: Conv+ReLU, then many Conv+BatchNorm+ReLU blocks, then a final Conv (Figure 5.4; this is the network we met in the batch-normalization lecture).

On the *castle* image at $\sigma = 50$ (input 14.76 dB) the slides report BM3D 26.21 dB, WNNM 26.51 dB, EPLL 26.36 dB and DnCNN 26.90–26.92 dB. The gains were real but modest, and on texture-rich images early supervised networks did not consistently beat BM3D, which led to some pessimism at the time about learned denoisers.

The lecture asked the class what Equation 5.5 reminds us of. A network that takes a noisy image and outputs the noise component is exactly the ϵ-prediction network of a diffusion model (Ho et al. 2020). The difference is in what the network is used for: DnCNN is trained for one noise level (DnCNN-S) or a range of levels (the blind model DnCNN-B, $\sigma \in [0, 55]$) and is applied once, whereas a diffusion model is trained across a whole schedule of noise levels and is applied repeatedly to generate samples. When DnCNN appeared it was not clear why estimating the noise should be the right thing to do; Tweedie's formula, below, gives the answer.

5.4.3. When clean targets do not exist

The weakness of Equation 5.4 is in its data: it requires matched clean images. In many scientific problems these do not exist.

- In the black-hole image of the Event Horizon Telescope, nobody has ever seen the object, so there is no ground truth at all.
- In cryo-electron microscopy (cryo-EM), the electron dose must be kept low to avoid destroying the specimen, so every image is very noisy (input SNR around 0 dB) and a clean version cannot be acquired.
- In live-cell or fast-moving imaging, the object moves between exposures, so even long exposures or averaging do not produce a clean reference.

Simulated noise is no escape, because real physical noise rarely matches the simulation. This motivates the question of the lecture: *how can we train a denoiser without matched clean images?*

5.5. Noise2Noise: learning from pairs of noisy images

5.5.1. The surprising observation

Noise2Noise (Lehtinen et al. 2018) (slides 12–13) replaces the clean target by a *second* noisy observation of the same image. With two measurements

$$y_1 = x + w_1, \qquad y_2 = x + w_2,$$

the network is trained to map one to the other (Figure 5.5, top):

$$\min_{\Theta} \mathbb{E}\,\|y_2 - F_\Theta(y_1)\|^2. \tag{5.6}$$

One would expect the network to learn to reproduce noise, since that is what the targets contain. Instead, at test time it outputs a clean image, and the quality is almost that of supervised training: in the slides' text-removal example the network trained on corrupted targets reaches 35.75 dB against 35.82 dB with clean targets.

5.5.2. Why the minimizer does not change

Theorem 5.1 (Noise2Noise has the supervised minimizer)**.** *Assume the second noise is conditionally zero-mean given the input, $\mathbb{E}[w_2 \mid x, y_1] = 0$, and that all second moments are finite. (This holds, in particular, when w_2 is independent of (x, w_1) and $\mathbb{E}[w_2] = 0$.) Then for every F,*

$$\mathbb{E}\|y_2 - F(y_1)\|^2 = \mathbb{E}\|x - F(y_1)\|^2 + \mathbb{E}\|w_2\|^2, \tag{5.7}$$

so the Noise2Noise loss and the supervised loss differ by a constant that does not depend on F, and both are minimized by $F^*(y_1) = \mathbb{E}[x \mid y_1]$.

Proof. Substitute $y_2 = x + w_2$ and expand the square:

$$\mathbb{E}\|x + w_2 - F(y_1)\|^2 = \mathbb{E}\|x - F(y_1)\|^2 + \mathbb{E}\|w_2\|^2 + 2\,\mathbb{E}\big[(x - F(y_1))^\top w_2\big].$$

For the cross term, condition on (x, y_1). The vector $x - F(y_1)$ is then fixed, so

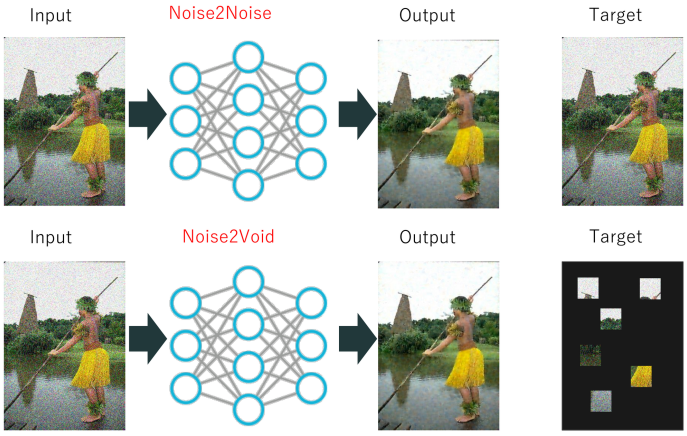


Figure 5.5.: Training targets of the self-supervised methods, as drawn in the lecture. Top: Noise2Noise feeds one noisy image to the network and uses a second, independently corrupted image of the same scene as the target. Bottom: Noise2Void uses a single noisy image; the target consists only of a few masked pixels of that same image (the bright squares), and the loss is computed there only. In both cases the output is a clean image although no clean image is ever shown. From slides 12 and 14 of the course slides (Ye 2026c).

$$\mathbb{E}[(x - F(y_1))^\top w_2] = \mathbb{E}[(x - F(y_1))^\top \mathbb{E}[w_2 \mid x, y_1]] = 0.$$

This proves Equation 5.7. The minimizer of the right-hand side is $\mathbb{E}[x \mid y_1]$ by Lemma 5.1. □

Two remarks make the statement precise.

- **What is needed is only that the noise in the target averages to zero given the input.** The noise need not be Gaussian, need not be independent across pixels, and need not have the same distribution as w_1. In particular, $\mathbb{E}[y_2 \mid y_1] = \mathbb{E}[x \mid y_1]$: the network regresses on the noisy target, and the regression function of the noisy target coincides with that of the clean one.
- **The price is variance, not bias.** With finitely many training pairs, the noisy targets make the empirical loss a noisier estimate of the population loss, so more data are needed for the same accuracy.

5.5.3. Which loss recovers which statistic

The squared loss recovers the mean of the target distribution. Other losses recover other statistics, and Noise2Noise exploits this to handle noise that is *not* zero-mean. For a scalar random variable z consider $c^* = \arg\min_c \mathbb{E}|z - c|^p$.

- L_2 **($p = 2$):** $\frac{d}{dc}\mathbb{E}(z-c)^2 = -2(\mathbb{E}z - c) = 0$ gives $c^* = \mathbb{E}z$, the mean.
- L_1 **($p = 1$):** for a continuous z with CDF G, $\mathbb{E}|z-c| = \int_{-\infty}^{c}(c-t)\,dG(t) + \int_{c}^{\infty}(t-c)\,dG(t)$, so $\frac{d}{dc}\mathbb{E}|z-c| = G(c) - (1-G(c)) = 2G(c) - 1$. Setting it to zero gives $G(c^*) = 1/2$, the median.
- L_0 **(mode seeking):** for a discrete target the zero-one loss gives $\mathbb{E}[\mathbf{1}\{z \neq c\}] = 1 - \Pr(z = c)$, so its minimizer is the most probable value, the mode. For a continuous target one maximizes instead $\Pr(|z-c| \leq h)$ for a small window h, which approaches a mode of the density as $h \to 0$ under regularity conditions. Noise2Noise implements this with the surrogate $(|z-c| + \epsilon)^p$ and p annealed from 2 toward 0 during training; this is a practical mode-seeking device, not a theorem that the minimizer reaches the mode for every distribution.

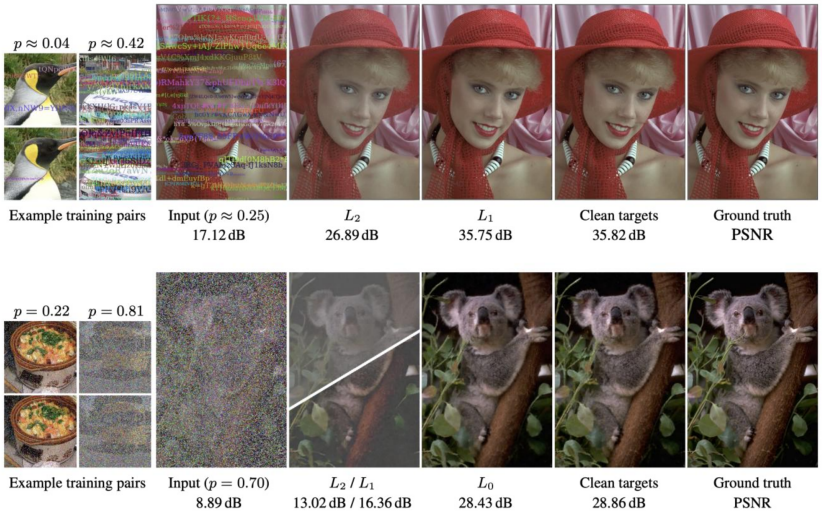


Figure 5.6.: Noise2Noise with non-zero-mean corruptions. Top: random text overlays; the network is trained only on pairs of differently corrupted images (left). The L_2 loss leaves a shadow of the mean text colour, while L_1 recovers the median and almost matches training with clean targets. Bottom: random-valued impulse noise with $p = 0.70$; only the annealed L_0 loss recovers the clean image. PSNR values are under each panel. From slide 13 of the course slides (Ye 2026c), originally from Lehtinen et al. (2018).

The text-removal and impulse-noise examples on slide 13 (Figure 5.6) show why this matters. A random text overlay replaces a fraction p of pixels by text colours; the corruption is not zero-mean, so the L_2 loss converges to the average of the clean colour and the text colours and leaves a visible shadow (26.89 dB at $p \approx 0.25$). When the clean colour is (nearly) determined by the input and $p < 1/2$, it is still the *median* of the target distribution at each pixel, so the L_1 loss recovers it (35.75 dB, against 35.82 dB with clean targets). For random-valued impulse noise at $p = 0.70$, even the median is corrupted (L_2/L_1 give 13.02/16.36 dB); but the clean value is still the single most likely

value, so the annealed L_0 loss recovers the mode and reaches 28.43 dB, against 28.86 dB with clean targets.

One caution: the clean-target equivalence of Theorem 5.1 was proved for the squared loss only. For L_1 there is no general analogue, because adding independent zero-median noise to an *uncertain* clean value does not preserve its conditional median. The L_1 and L_0 results rely on the corruption leaving the relevant statistic of the target unchanged, which has to be checked for each corruption model.

5.5.4. The limitation

Noise2Noise removes the need for clean images, but it still needs *two independent noisy observations of exactly the same scene*. For a moving object, a dose-limited specimen or a one-off astronomical event, that is usually as hard to get as a clean image. The next question is therefore: can we train from a *single* noisy image?

5.6. Noise2Self, J-invariance and Noise2Void

5.6.1. Why a single noisy image is a problem

The obvious loss with a single image is the autoencoder loss $\mathbb{E}\|y - F_\Theta(y)\|^2$. It has a trivial minimizer: the identity $F(y) = y$ achieves zero loss and removes no noise at all. Any single-image method must exclude the identity. Two ways are known: restrict the network so that it *cannot* see the pixel it predicts (this section), or add a penalty that makes copying expensive (SURE, next section).

5.6.2. J-invariance

The slides present this idea through Noise2Void; the formal argument below, from Noise2Self (Batson and Royer 2019), is added here because it explains why hiding a pixel rules out the identity. Let $\mathcal{J} = \{J_1, \dots, J_K\}$ be a partition of the pixel indices $\{1, \dots, n\}$. For a subset J write y_J for the pixels in J and J^c for its complement.

Definition 5.1 (J-invariant function)**.** A function $F : \mathbb{R}^n \to \mathbb{R}^n$ is *$\mathcal{J}$-invariant* if for every $J \in \mathcal{J}$ the output $F(y)_J$ does not depend on the input y_J; that is, $F(y)_J$ is a function of y_{J^c} only.

The identity is not $\mathcal{J}$-invariant, since $F(y)_J = y_J$ depends on nothing but y_J. The point of the definition is the following identity.

Theorem 5.2 (Self-supervised loss for J-invariant functions)**.** *Suppose that, given x, the noisy pixels in different subsets are independent (y_J is conditionally independent of y_{J^c} given x, for each $J \in \mathcal{J}$), that the noise is zero-mean, $\mathbb{E}[y \mid x] = x$, and that second moments are finite. Then for every $\mathcal{J}$-invariant F,*

$$\mathbb{E}\|F(y) - y\|^2 = \mathbb{E}\|F(y) - x\|^2 + \mathbb{E}\|y - x\|^2. \tag{5.8}$$

Proof. Expand $\|F(y) - y\|^2 = \|(F(y) - x) - (y - x)\|^2$:

$$\mathbb{E}\|F(y) - y\|^2 = \mathbb{E}\|F(y) - x\|^2 + \mathbb{E}\|y - x\|^2 - 2\sum_{J \in \mathcal{J}} \mathbb{E}\big[(F(y)_J - x_J)^\top (y_J - x_J)\big].$$

Fix one J and condition on x. By $\mathcal{J}$-invariance, $F(y)_J - x_J$ is a function of (y_{J^c}, x) only; by assumption y_{J^c} and y_J are independent given x. The conditional expectation of the product therefore factorizes:

$$\mathbb{E}\big[(F(y)_J - x_J)^\top (y_J - x_J) \,\big|\, x\big] = \mathbb{E}\big[F(y)_J - x_J \,\big|\, x\big]^\top \mathbb{E}\big[y_J - x_J \,\big|\, x\big] = 0,$$

because $\mathbb{E}[y_J \mid x] = x_J$. Taking the outer expectation removes every cross term. □

So, *within the class of $\mathcal{J}$-invariant functions*, minimizing the self-supervised loss on the noisy image is the same as minimizing the true error against the unseen clean image. Noise2Noise used independence *between two images*; Noise2Self uses independence *between pixels of one image*.

Note what the restriction costs. The best $\mathcal{J}$-invariant predictor of x_J is $\mathbb{E}[x_J \mid y_{J^c}]$, not $\mathbb{E}[x_J \mid y]$: the network never uses the noisy value of the pixel it is predicting, which still carries information about that pixel. This is one reason blind-spot methods usually trail supervised training by a small margin.

5.6.3. Noise2Void: the blind-spot network in practice

Noise2Void (Krull et al. 2019) (slides 14–16) arrived at the same idea from the practical side, in the setting of microscopy. In a conventional CNN the receptive field of an output pixel contains the corresponding input pixel; a *blind-spot network* excludes it (Figure 5.7, left). Rather than designing such an architecture, Noise2Void trains an ordinary CNN with a masking scheme (Figure 5.7, right; target of Figure 5.5, bottom):

1. pick a random set M of pixels in a training patch;
2. replace each y_i, $i \in M$, by the value of a randomly chosen neighbour *other than i itself*, giving a modified input $\tilde{y}$;
3. compute the loss only on the masked pixels: $\sum_{i \in M} (F_\Theta(\tilde{y})_i - y_i)^2$.

Because the network never sees the original y_i at a masked location, it cannot copy it, and the argument of Theorem 5.2 applies to the masked pixels. In the lecture's exaggerated picture, the network is shown an image with holes and learns to fill the holes from the context; because the noise in a hole is independent of its context, the best it can do is predict the clean value. One caveat: at test time the image is fed *without* masks, so the guarantee of Theorem 5.2 describes training; only an architectural blind spot keeps $\mathcal{J}$-invariance at inference.

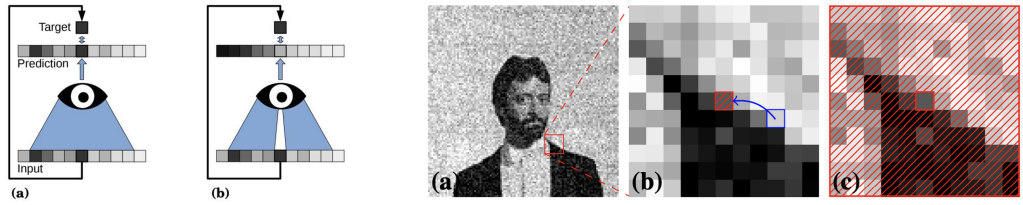


Figure 5.7.: The blind spot in the original Noise2Void paper. Left, (a) and (b): in a conventional network the prediction for a pixel depends on a patch of input pixels that includes the pixel itself (blue cone), so training with the same noisy image as input and target yields the identity; a blind-spot network excludes the pixel from its own receptive field. Right, (a) to (c): the masking scheme used in training. In a patch (b) of the noisy image (a), a randomly selected pixel (blue) is copied into the masked location (red); the target (c) is the unmodified noisy patch, and the loss is computed only at the masked pixels. From slide 15 of the course slides (Ye 2026c), originally Fig. 2 and 3 of Krull et al. (2019).

The slides compare Noise2Void with BM3D, supervised training ("traditional") and Noise2Noise. On BSD68 the PSNRs are 28.59 (BM3D), 29.06 (supervised), 28.86 (Noise2Noise) and 27.71 dB (Noise2Void); on simulated cell-membrane data they are 29.96, 32.56, 32.43 and 32.28 dB. Noise2Void gives up a little accuracy, but it needs neither clean nor paired data. On real cryo-TEM and fluorescence data, where no clean target exists and often no second noisy copy either, only BM3D and Noise2Void are applicable at all, and Noise2Void is much faster at inference (about 0.1–1.3 s against 4.6–33.2 s for BM3D on the slides' examples).

A student asked how the model knows which part of the image is noise. The answer is that it does not need to: the model assumes every pixel is signal plus noise, and the independence of the noise is the only thing that distinguishes the two. *Under that independence assumption*, whatever the network can predict from the context must come from the signal, because the noise at a pixel carries no information about its neighbours. When the noise is spatially correlated this breaks down, and the network also predicts the correlated part of the noise (see Section 5.12).

The lecture also asked what masking pixels and predicting them reminds us of. It is masked token prediction, as in masked language modelling (and, at the level of

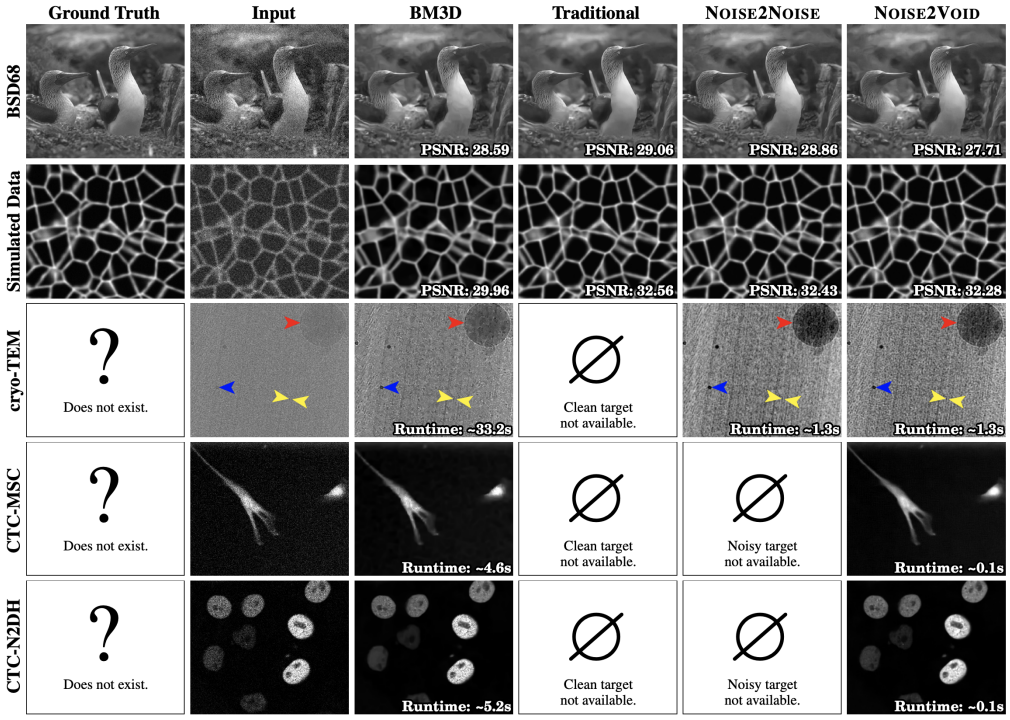


Figure 5.8.: Noise2Void against BM3D, supervised training ("traditional") and Noise2Noise, on the five datasets of the lecture. The top two rows have a ground truth and are scored by PSNR. The bottom three — cryo-TEM and two fluorescence-microscopy sets — have none: the ground-truth column reads "does not exist", supervised training has no clean target, and on two of the three Noise2Noise has no second noisy copy either, so only BM3D and Noise2Void produce an image at all. These are the cases the method exists for, and they are the ones a PSNR table cannot report. From slide 16 of the course slides (Ye 2026c), originally from Krull et al. (2019).

representations, JEPA). At the end of the lecture this resemblance is revisited as an analogy: both predict hidden content from its context, which for continuous images is tied to the score, while for discrete tokens it is an analogy rather than a theorem.

5.7. SURE: a single image with a divergence penalty

5.7.1. Stein's unbiased risk estimate

The second way to exclude the identity keeps the plain autoencoder loss and adds a penalty. For Gaussian noise $w \sim \mathcal{N}(0, \sigma^2 I)$, the SURE denoiser (Soltanayev and Chun 2018) (slides 17–19) minimizes

$$\ell_{SURE}(\Theta) = \mathbb{E}_{y\sim P_Y}\, d(y, F_\Theta(y)), \qquad d(y, F_\Theta(y)) := \|y - F_\Theta(y)\|^2 + 2\sigma^2 \operatorname{div}_y F_\Theta(y). \tag{5.9}$$

The first term is the autoencoder loss; the second, the *divergence-based penalty*, is large for the identity, whose divergence is n. The construction comes from classical statistics, where Stein's unbiased risk estimator (SURE) (Stein 1981) was used for decades, for example to tune wavelet thresholds, before deep learning. The key fact is Stein's lemma.

Lemma 5.2 (Stein's lemma)**.** *Let $w \sim \mathcal{N}(0, \sigma^2 I_n)$ and let $g : \mathbb{R}^n \to \mathbb{R}^n$ be weakly differentiable with $\mathbb{E}|\partial g_i/\partial y_i| < \infty$. Then, for fixed x and $y = x + w$,*

$$\mathbb{E}[w^\top g(y)] = \sigma^2\, \mathbb{E}[\operatorname{div}_y g(y)].$$

Proof. It suffices to prove the scalar identity $\mathbb{E}[w_i\, g_i(x + w)] = \sigma^2\, \mathbb{E}[\partial_i g_i(x + w)]$ for each coordinate and sum over i. Fix all coordinates except w_i; the density of w_i is $\phi(t) = (2\pi\sigma^2)^{-1/2} e^{-t^2/2\sigma^2}$, which satisfies $t\,\phi(t) = -\sigma^2 \phi'(t)$. Integrating by parts,

$$\int t\,\phi(t)\, g_i(\dots, x_i + t, \dots)\, dt = -\sigma^2 \Big[\phi(t)\, g_i\Big]_{-\infty}^{\infty} + \sigma^2 \int \phi(t)\, \partial_i g_i(\dots, x_i + t, \dots)\, dt.$$

The boundary term vanishes under mild growth conditions (for example, if g_i grows at most polynomially; Stein (1981) gives the precise conditions). Taking the expectation over the other coordinates gives the claim. □

Theorem 5.3 (SURE is unbiased for the true error)**.** *Under the assumptions of Lemma 5.2, and if $\mathbb{E}\|F(y)\|^2 < \infty$, for any fixed x,*

$$\mathbb{E}\|F(y) - x\|^2 = \mathbb{E}\Big[\|y - F(y)\|^2 + 2\sigma^2 \operatorname{div}_y F(y)\Big] - n\sigma^2. \tag{5.10}$$

Proof. Write $F(y) - x = (F(y) - y) + w$ and expand:

$$\mathbb{E}\|F(y) - x\|^2 = \mathbb{E}\|F(y) - y\|^2 + 2\,\mathbb{E}\big[w^\top (F(y) - y)\big] + \mathbb{E}\|w\|^2.$$

The last term is $n\sigma^2$. For the middle term, $\mathbb{E}[w^\top y] = \mathbb{E}[w^\top x] + \mathbb{E}\|w\|^2 = n\sigma^2$, and $\mathbb{E}[w^\top F(y)] = \sigma^2\,\mathbb{E}[\mathrm{div}_y F(y)]$ by Stein's lemma. Hence the middle term is $2\sigma^2\mathbb{E}[\mathrm{div}_y F] - 2n\sigma^2$, and collecting terms gives Equation 5.10. □

As a sanity check, the identity $F(y) = y$ has zero autoencoder loss but divergence n, so its SURE value is $2n\sigma^2 - n\sigma^2 = n\sigma^2$, exactly the true error of not denoising: the penalty removes the identity's artificial advantage. The right-hand side of Equation 5.10 can be evaluated *without* x. It is unbiased only in expectation; for one image it can even be negative, and a very flexible network can overfit it on a small dataset. The constant $-n\sigma^2$ does not affect the minimizer, which is why Equation 5.9 drops it. Because the identity holds for every fixed x, it also holds after averaging over x, so minimizing Equation 5.9 over networks is minimizing the supervised risk.

5.7.2. Computing the divergence: Monte-Carlo SURE

For a network with millions of outputs, $\mathrm{div}_y F_\Theta$ is the trace of an $n \times n$ Jacobian and cannot be formed explicitly. The SURE denoiser uses the Monte-Carlo estimate of Ramani et al. (2008): draw $\tilde{n} \sim \mathcal{N}(0, I)$ independent of y and use, for a small $\epsilon > 0$,

$$\mathrm{div}_y F(y) \approx \frac{1}{\epsilon}\,\tilde{n}^\top (F(y + \epsilon\tilde{n}) - F(y)). \tag{5.11}$$

To see why, Taylor-expand $F(y + \epsilon\tilde{n}) - F(y) = \epsilon\, J_F(y)\,\tilde{n} + O(\epsilon^2)$, where J_F is the Jacobian. Then the right-hand side of Equation 5.11 equals $\tilde{n}^\top J_F(y)\,\tilde{n} + O(\epsilon)$, and since $\mathbb{E}[\tilde{n}\tilde{n}^\top] = I$,

$$\mathbb{E}_{\tilde{n}}\big[\tilde{n}^\top J_F(y)\,\tilde{n}\big] = \mathrm{tr}\,\big(J_F(y)\,\mathbb{E}[\tilde{n}\tilde{n}^\top]\big) = \mathrm{tr}\,J_F(y) = \mathrm{div}_y F(y).$$

One extra forward pass per training step is enough; the randomness of $\tilde{n}$ averages out over training like the randomness of a minibatch. Figure 5.14 checks both Equation 5.10 and Equation 5.11 numerically.

On one BSD68 image at $\sigma = 50$ (input 14.76 dB) the slides report BM3D 26.14 dB, a SURE-trained network 26.46 dB, and the same network trained with clean targets (MSE) 26.85 dB: SURE does not reach supervised quality, but it is close and it needs no clean data. Figure 5.9 is that comparison.



(a) Noisy image / 14.76dB (b) BM3D / 26.14dB (c) SURE / 26.46dB (d) SURE-T / 26.46dB (e) MSE / 26.85dB

Figure 5: Denoising results of an image from the BSD68 dataset for σ=50

Figure 5.9.: SURE on a natural image: BSD68 at $\sigma = 50$, with the noisy input at 14.76 dB. (c) is the network trained with the SURE loss and no clean data, (d) the same with a transformed divergence estimate, and (e) the same architecture trained on clean targets. The insets show that the 0.4 dB separating (c) from (e) is visible mainly in fine texture. From slide 19 of the course slides (Ye 2026c), originally Fig. 5 of Soltanayev and Chun (2018).

5.8. A side-by-side comparison

Slide 20 puts the three approaches next to each other.

- **Supervised learning:** $\ell_{sup}(\Theta) = \|x - F_\Theta(y)\|^2$, with noisy-clean pairs (x, y).
- **Noise2X** (Noise2Noise, Noise2Self, Noise2Void, …): $\ell_{selfsup}(\Theta) = \|z - F_\Theta(y)\|^2$, where the target z is derived from the noisy data in a way specific to each algorithm: a second noisy copy, the masked pixels, or (in Noisier2Noise (Moran et al. 2020)) the original y when the input is an even noisier version of it.
- **SURE:** $\|y - F_\Theta(y)\|^2 + 2\sigma^2 \operatorname{div}_y F_\Theta(y)$, a divergence penalty compensating for the use of y itself as target.

Written this way, the losses differ mainly in the target (x, z or y), together with what each method adds to make that target usable: an independent second copy, a restriction on what the network may see, or the divergence penalty. These algorithmic choices look heuristic and unrelated, and each comes with its own proof. Is there a *unified mathematical theory* of self-supervised denoising — one that explains why all of them work and points to something better? The answer of Kim and Ye (2021) is Tweedie's formula.

5.9. Tweedie's formula

5.9.1. Exponential families and their posteriors

Following slides 22–23, a family of densities $p(y \mid \eta)$ indexed by a parameter η is an **exponential family** if it can be written as

$$p(y \mid \eta) = p_0(y)\, \exp(\eta^\top T(y) - \varphi(\eta)), \tag{5.12}$$

where $p_0(y)$ depends on y only, $T(y)$ is the *sufficient statistic*, η is the *natural parameter*, and $\varphi(\eta)$ depends on η only and makes the density integrate to one. The parameter enters only through the cross term $\eta^\top T(y)$. Gaussian, Poisson, Gamma, Bernoulli and exponential distributions are all of this form, as the table in Section 5.9.6 shows. The decomposition is not unique (we will see two for the Gaussian), but the result below does not depend on the choice.

Put a prior $p(\eta)$ on the parameter. Bayes' rule gives the posterior

$$p(\eta \mid y) = \frac{p(y \mid \eta)\, p(\eta)}{p(y)} = \frac{p_0(y)}{p(y)}\, \exp(\eta^\top T(y))\, e^{-\varphi(\eta)}\, p(\eta),$$

and absorbing $p_0(y)/p(y)$ into the exponent,

$$p(\eta \mid y) = \exp(\eta^\top T(y) - \lambda(y))\, [p(\eta)\, e^{-\varphi(\eta)}], \qquad \lambda(y) = \log \frac{p(y)}{p_0(y)}, \tag{5.13}$$

where $p(y) = \int p(y \mid \eta) p(\eta)\, d\eta$ is the marginal density of the data. The posterior is itself an exponential family, now in η, with the roles of parameter and statistic exchanged: the cross term is again $\eta^\top T(y)$, and the bracket depends only on η.

5.9.2. The closed form of the posterior mean

Under squared loss the Bayes-optimal estimate of η is the posterior mean $\hat{\eta} = \mathbb{E}[\eta \mid y]$ (Lemma 5.1). The remarkable fact, due to Robbins and Tweedie and popularized by Efron (2011), is that this posterior mean can be written in terms of the marginal density of y alone.

Theorem 5.4 (Tweedie's formula for exponential families)**.** *Let y be scalar, let $p(y \mid \eta)$ have the form Equation 5.12 with T differentiable, and assume $p(y)$ and $p_0(y)$ are positive and differentiable and that differentiation in y may be exchanged with integration in η. Then*

$$\hat{\eta}^\top T'(y) = \lambda'(y) = -\nabla_y \log p_0(y) + \nabla_y \log p(y) = -l_0'(y) + l'(y), \tag{5.14}$$

where $\hat{\eta} = \mathbb{E}[\eta \mid y]$ *and* $T'(y)$ *is the derivative of the sufficient statistic. The quantity* $l'(y) = \nabla_y \log p(y)$ *is the* ***score function*** *of the noisy data.*

Proof. The posterior Equation 5.13 integrates to one for every y:

$$\int \exp(\eta^\top T(y) - \lambda(y))\, p(\eta)\, e^{-\varphi(\eta)}\, d\eta = 1.$$

Differentiate both sides with respect to y. The bracket $p(\eta)e^{-\varphi(\eta)}$ does not depend on y, and the derivative of the exponent is $\eta^\top T'(y) - \lambda'(y)$, so

$$\int (\eta^\top T'(y) - \lambda'(y))\, p(\eta \mid y)\, d\eta = 0.$$

The first part of the integral is $\mathbb{E}[\eta \mid y]^\top T'(y)$; the second is $\lambda'(y)$ because $p(\eta \mid y)$ integrates to one. Hence $\hat{\eta}^\top T'(y) = \lambda'(y)$. Finally, $\lambda(y) = \log p(y) - \log p_0(y) = l(y) - l_0(y)$, which gives the right-hand side of Equation 5.14. □

Read Equation 5.14 from right to left. The noise model fixes T and p_0, which are known. The only thing that depends on the unknown clean-image distribution is the score $l'(y)$ of the *noisy* data — and the noisy data are exactly what we have. If we can estimate the score from noisy images, we obtain the Bayes-optimal estimate of η without ever seeing a clean image.

For an image $y \in \mathbb{R}^n$ with $T(y) \in \mathbb{R}^k$, the same proof differentiates with respect to each y_i. Writing $J_T(y) \in \mathbb{R}^{k \times n}$ for the Jacobian of T, the result is

$$J_T(y)^\top\, \mathbb{E}[\eta \mid y] = \nabla_y \log p(y) - \nabla_y \log p_0(y). \tag{5.15}$$

When the noise acts independently on each pixel, J_T is (block) diagonal and this is the scalar formula applied pixel by pixel; the score $\nabla_y \log p(y)$, however, still depends on the whole image, because the clean-image prior does not factorize. Two cautions follow from the form of Equation 5.15. The score determines $\mathbb{E}[\eta \mid y]$ only through the linear map $J_T^\top$, which sends $\mathbb{R}^k$ to $\mathbb{R}^n$; recovering an unrestricted k-dimensional $\mathbb{E}[\eta \mid y]$ needs this map to be injective, i.e. $\operatorname{rank} J_T = k$ (J_T has full *row* rank). Known components of η can supply the missing information, as for $\eta_1 = \alpha - 1$ in the Gamma case below. And $\mathbb{E}[\eta(x) \mid y]$ is in general not $\eta(\mathbb{E}[x \mid y])$ when η is a nonlinear function of x; we meet both points in the Poisson and Gamma cases below.

5.9.3. Case 1: Gaussian noise

Let $y = x + w$ with $w \sim \mathcal{N}(0, \sigma^2)$ and σ known (slides 24–25). Expanding the square in the Gaussian likelihood separates the y-only, cross and x-only terms:

$$p(y \mid x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-x)^2}{2\sigma^2}} = \underbrace{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}}_{p_0(y)} \exp\Big(\underbrace{\frac{x}{\sigma^2}}_{\eta} \underbrace{y}_{T(y)} - \underbrace{\frac{x^2}{2\sigma^2}}_{\varphi(\eta)} \Big).$$

So $\eta = x/\sigma^2$, $T(y) = y$ with $T'(y) = 1$, and $l_0(y) = -\frac{y^2}{2\sigma^2} + \text{const}$, whence $-l_0'(y) = y/\sigma^2$. Substituting into Equation 5.14,

$$\mathbb{E}[\eta \mid y] = \frac{\hat{x}}{\sigma^2} = \frac{y}{\sigma^2} + l'(y),$$

and multiplying by σ^2,

$$\boxed{\hat{x} = \mathbb{E}[x \mid y] = y + \sigma^2\, l'(y), \qquad l'(y) = \nabla_y \log p(y).} \tag{5.16}$$

For an image with $w \sim \mathcal{N}(0, \sigma^2 I)$ the same computation holds pixel by pixel, with $l'(y) = \nabla_y \log p(y) \in \mathbb{R}^n$. More generally, for correlated Gaussian noise $w \sim \mathcal{N}(0, \Sigma)$ the vector decomposition $\eta = \Sigma^{-1}x$, $T(y) = y$, $l_0(y) = -\frac{1}{2}y^\top \Sigma^{-1} y + \text{const}$ gives $\Sigma^{-1}\hat{x} = \Sigma^{-1}y + \nabla_y \log p(y)$, that is $\hat{x} = y + \Sigma\, \nabla_y \log p(y)$; we return to this when we discuss correlated noise.

The decomposition is not unique. The second Gaussian row of the table uses $\eta = [x/\sigma^2, -1/(2\sigma^2)]^\top$, $T(y) = [y, y^2]^\top$ and $p_0(y) = 1/\sqrt{2\pi}$, so that $l_0' = 0$ and $T'(y) = [1, 2y]^\top$. Then Equation 5.14 reads $\hat{x}/\sigma^2 - 2y/(2\sigma^2) = l'(y)$, which is again Equation 5.16. The term that was in p_0 has moved into $\eta^\top T$, and the answer is unchanged.

Denoising is score estimation. Equation 5.16 can be read in both directions:

$$l'(y) = \frac{\mathbb{E}[x \mid y] - y}{\sigma^2}, \qquad \mathbb{E}[w \mid y] = y - \mathbb{E}[x \mid y] = -\sigma^2\, l'(y). \tag{5.17}$$

An MMSE denoiser and the score of the noisy distribution determine each other. The second identity answers the question left open by DnCNN: a network trained to predict the noise from y converges to $\mathbb{E}[w \mid y]$, which is the score scaled by $-\sigma^2$. Predicting noise, predicting the clean image and predicting the score are the same task. Figure 5.13 verifies Equation 5.16 on a two-component Gaussian mixture.

A direct check without exponential families. In the Gaussian case the formula can also be verified by differentiating the marginal $p(y) = \int \mathcal{N}(y; x, \sigma^2 I)\, p(x)\, dx$ under the integral. Since $\nabla_y \mathcal{N}(y; x, \sigma^2 I) = -\frac{y-x}{\sigma^2}\, \mathcal{N}(y; x, \sigma^2 I)$,

$$\nabla_y p(y) = \int \frac{x-y}{\sigma^2}\,\mathcal{N}(y;x,\sigma^2 I)\,p(x)\,dx \quad\Longrightarrow\quad \nabla_y \log p(y) = \frac{\mathbb{E}[x \mid y]-y}{\sigma^2},$$

after dividing by $p(y)$ and recognising the posterior $p(x \mid y)$ inside the integral. This is Equation 5.17 again, in any dimension.

A simple closed-form case. If $x \sim \mathcal{N}(\mu,\tau^2)$, then $y \sim \mathcal{N}(\mu,\tau^2+\sigma^2)$ and $l'(y) = -(y-\mu)/(\tau^2+\sigma^2)$, so Tweedie gives $\hat{x} = \frac{\tau^2}{\tau^2+\sigma^2}y + \frac{\sigma^2}{\tau^2+\sigma^2}\mu$, the familiar Wiener shrinkage. With $\mu=0$, $\tau^2=4$, $\sigma^2=1$ and $y=3$, the score is -0.6 and $\hat{x} = 3+1\cdot(-0.6) = 2.4$; the Bayes risk is $\tau^2\sigma^2/(\tau^2+\sigma^2) = 0.8$ against 1 for the raw observation.

5.9.4. Case 2: Poisson noise

Photon-limited imaging (low-light fluorescence, astronomy, PET; slide 26) produces counts: $y \in \{0,1,2,\dots\}$ with mean $x>0$,

$$p(y \mid x) = \frac{x^y e^{-x}}{y!} = \underbrace{\frac{1}{y!}}_{p_0(y)}\,\exp\!\big(\underbrace{\log x}_{\eta}\cdot\underbrace{y}_{T(y)} - \underbrace{x}_{\varphi(\eta)=e^\eta}\big).$$

Here $\eta = \log x$, $T(y)=y$ and $T'(y)=1$. To differentiate in y we extend $p(y)$ and $p_0(y)$ smoothly to real y, using $y! = \Gamma(y+1)$. Then $l_0(y) = -\log\Gamma(y+1)$ and $l_0'(y) = -\psi(y+1)$, where ψ is the digamma function. The asymptotic expansion $\psi(z) = \log z - \frac{1}{2z} + O(z^{-2})$ gives $\psi(y+1) = \log(y+\frac{1}{2}) + O(y^{-2})$, which is the approximation on the slides:

$$l_0'(y) \approx -\log(y+\tfrac{1}{2}).$$

Substituting into Equation 5.14 gives the exact relation and its approximation,

$$\mathbb{E}[\log x \mid y] = \psi(y+1) + l'(y) \approx \log(y+\tfrac{1}{2}) + l'(y),$$

and writing $\log\hat{x}$ for the right-hand side,

$$\boxed{\hat{x} = (y+\tfrac{1}{2})\exp(l'(y)).} \tag{5.18}$$

One subtlety deserves to be stated. Tweedie's formula gives the posterior mean of the *natural parameter*, here $\log x$. So Equation 5.18 is (approximately) $\exp(\mathbb{E}[\log x \mid y])$, the posterior *geometric* mean, which by Jensen's inequality is never larger than $\mathbb{E}[x \mid y]$; the digamma approximation adds a further error at small counts. The worked example of Section 5.15.1.4 measures both gaps.

For comparison, the exact posterior mean of x also has a closed form in terms of the marginal, without any derivative (Robbins' formula; see Efron (2011)). Since $x \cdot \frac{x^y e^{-x}}{y!} = (y+1)\frac{x^{y+1}e^{-x}}{(y+1)!}$,

$$(y+1)\,p(y+1) = \int x\,\frac{x^y e^{-x}}{y!}\,p(x)\,dx \quad\Longrightarrow\quad \mathbb{E}[x \mid y] = (y+1)\,\frac{p(y+1)}{p(y)}.$$

For counts, a ratio of neighbouring probabilities plays the role that the score plays for continuous data. Note that the two formulas estimate different quantities: Robbins' ratio gives the arithmetic posterior mean exactly, whereas Equation 5.18 approximates the geometric one (compare the columns of Table 5.5 below).

5.9.5. Case 3: Gamma noise

Multiplicative noise — speckle in ultrasound and synthetic aperture radar, for instance — is modelled by a Gamma distribution whose scale is proportional to the clean value. Following slide 27, write

$$p(y \mid x) \propto \left(\frac{y}{x}\right)^{\alpha-1} e^{-\beta\frac{y}{x}} = \exp\Big((\alpha-1)\log y - \frac{\beta}{x}\,y - (\alpha-1)\log x\Big),$$

with α, β known. This is the model $y = x\,n$ with multiplicative noise $n \sim \mathrm{Gamma}(\alpha, \beta)$ (shape α, rate β); the change of variables $n = y/x$ contributes a factor $1/x$, and the normalized density is

$$p(y \mid x) = \frac{\beta^\alpha}{\Gamma(\alpha)\,x^\alpha}\,y^{\alpha-1}e^{-\beta y/x}, \qquad y > 0,$$

with $\mathbb{E}[y \mid x] = \alpha x/\beta$. The noise has mean one when $\alpha = \beta = k$, which is the setting "Gamma ($k = 100$)" of the experiments below. The x-only factors are absorbed in $\varphi(\eta)$, which plays no role in Equation 5.14. The parameter enters through two cross terms, so η and T are vectors:

$$p_0(y) = 1, \qquad \eta = \begin{bmatrix}\alpha - 1\\ -\beta/x\end{bmatrix}, \qquad T(y) = \begin{bmatrix}\log y\\ y\end{bmatrix}, \qquad T'(y) = \begin{bmatrix}1/y\\ 1\end{bmatrix}.$$

(The summary table on the slides lists $T(y) = [\log y, -y]^\top$; with $\eta_2 = -\beta/x$, the sign that reproduces $e^{-\beta y/x}$ is $T_2(y) = y$, as on the derivation slide.) Since $l'_0 = 0$, Equation 5.14 gives

$$\frac{\alpha - 1}{y} - \beta\,\mathbb{E}\Big[\frac{1}{x}\,\Big|\,y\Big] = l'(y).$$

Writing $\beta/\hat{x}$ for $\beta\,\mathbb{E}[1/x \mid y]$ and solving for $\hat{x}$,

$$\boxed{\hat{x} = \frac{\beta y}{(\alpha - 1) - y\, l'(y)}.} \tag{5.19}$$

Here $\hat{x}$ is the posterior *harmonic* mean, $1/\mathbb{E}[1/x \mid y]$, again because the natural parameter is $-\beta/x$ rather than x. For an exact score the denominator $(\alpha-1)-y\,l'(y) = \beta y\,\mathbb{E}[1/x \mid y]$ is positive; with an estimated score it can approach zero or change sign, which an implementation has to guard against.

A conjugate example checks both the algebra and the interpretation. Take the prior $x \sim \text{InverseGamma}(a_0, b_0)$, with density proportional to $x^{-a_0-1}e^{-b_0/x}$. Multiplying by the likelihood gives the posterior $\text{InverseGamma}(a_0+\alpha,\, b_0+\beta y)$, and integrating x out gives $p(y) \propto y^{\alpha-1}(b_0+\beta y)^{-(a_0+\alpha)}$, so $l'(y) = \frac{\alpha-1}{y} - \frac{\beta(a_0+\alpha)}{b_0+\beta y}$. With $a_0 = 2$, $b_0 = 1$, $\alpha = \beta = 4$ and $y = 1$: $l'(1) = 3 - 4.8 = -1.8$, and Equation 5.19 gives $\hat{x} = 4/(3 + 1.8) = 5/6$. The posterior is $\text{InverseGamma}(6, 5)$, whose harmonic mean is $b/a = 5/6$ (exactly the formula) and whose mean is $b/(a-1) = 1$.

5.9.6. The full table

Repeating the computation for other members of the family gives the table of Kim and Ye (2021) (slide 28). It is split here into the decompositions (Table 5.1) and the resulting estimators (Table 5.2), with the Gamma sign corrected and the Gamma density normalized as above.

Table 5.1.: Exponential-family decompositions of common noise models.

Distribution	$p(y \mid x)$	η	$T(y)$	$p_0(y)$
Gaussian	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-x)^2}{2\sigma^2}}$	x/σ^2	y	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}$
Gaussian (alt.)	same	$[\frac{x}{\sigma^2}, -\frac{1}{2\sigma^2}]^\top$	$[y, y^2]^\top$	$\frac{1}{\sqrt{2\pi}}$
Poisson	$\frac{x^y e^{-x}}{y!}$	$\log x$	y	$\frac{1}{y!}$
Gamma(α, β)	$\frac{\beta^\alpha y^{\alpha-1}}{\Gamma(\alpha) x^\alpha} e^{-\beta y/x}$	$[\alpha - 1, -\frac{\beta}{x}]^\top$	$[\log y, y]^\top$	1
Bernoulli	$x^y(1-x)^{1-y}$	$\log \frac{x}{1-x}$	y	1
Exponential	xe^{-yx}, $y \geq 0$	$-x$	y	1

Table 5.2.: Tweedie denoisers obtained from Table 5.1 by solving $\hat{\eta}^\top T'(y) = -l_0'(y) + l'(y)$.

Distribution	$l_0'(y)$	Tweedie estimate $\hat{x}$	What $\hat{x}$ is
Gaussian	$-y/\sigma^2$	$y + \sigma^2 l'(y)$	posterior mean
Gaussian (alt.)	0	$y + \sigma^2 l'(y)$	posterior mean

Distribution	$l_0'(y)$	Tweedie estimate $\hat{x}$	What $\hat{x}$ is
Poisson	$\approx -\log(y+\frac{1}{2})$	$(y+\frac{1}{2})\exp(l'(y))$	approx. posterior geometric mean
Gamma(α,β)	0	$\frac{\beta y}{(\alpha-1)-y\,l'(y)}$	posterior harmonic mean
Bernoulli	0	$\frac{e^{l'(y)}}{1+e^{l'(y)}}$	logistic of posterior mean of η
Exponential	0	$-l'(y)$	posterior mean of the rate

The last two rows are one line each. For the exponential distribution, $p(y \mid x) = \exp(-x\,y + \log x)$, so $\eta = -x$, $T = y$, $p_0 = 1$, and Equation 5.14 gives $\mathbb{E}[-x \mid y] = l'(y)$, i.e. $\hat{x} = -l'(y)$; here $\hat{x}$ is the genuine posterior mean of x because η is linear in x. For the Bernoulli distribution, $\eta = \log \frac{x}{1-x}$ and $\hat{\eta} = l'(y)$; mapping back through the inverse of the logit gives the sigmoid in the table (which, as in the Poisson case, is the image of the posterior mean of η, not the posterior mean of x; for a binary y the derivative must again be understood through a smooth extension).

The message of the table, highlighted on the slides, is: **as long as we can compute the score function, optimal denoising can be achieved by Tweedie's formula**, for every noise model in the family. "Optimal" here has a precise meaning: the formula gives the posterior mean of the natural parameter, which is optimal for squared error *in that parameter*. It is the MMSE estimate of x itself when η is linear in x (Gaussian, exponential), and a geometric, harmonic or logistic posterior summary otherwise (Poisson, Gamma, Bernoulli).

5.10. Noise2Score

5.10.1. Training once, denoising by formula

Noise2Score (Kim and Ye 2021) (slide 29) turns the table into an algorithm with two phases.

1. **Training phase (noise-model agnostic).** Train a network R_Θ on noisy images only so that $R_{\Theta^*}(y) \approx l'(y) = \nabla_y \log p(y)$.
2. **Inference phase (noise-model specific).** Plug the estimated score into the Tweedie formula of Table 5.2 for the noise model at hand (Figure 5.10).

The noise model and its parameters (σ, or α and β) enter only in the second phase, and the training procedure is the same whatever the noise. This should not be read as "one trained network serves every noise model". The network learns the score of the observed distribution $p(y)$, and $p(y)$ depends on the noise: data with Poisson noise and

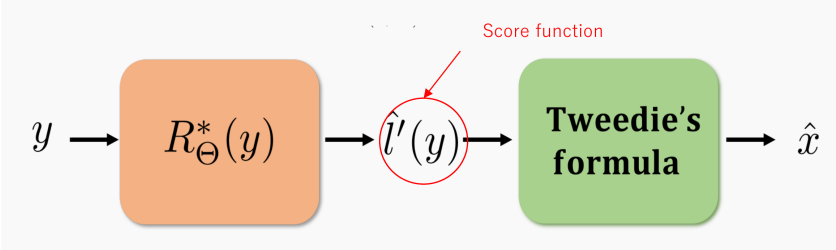


Figure 5.10.: The Noise2Score pipeline. A network R_{Θ^*} trained on noisy images outputs an estimate of the score $l'(y)$; Tweedie's formula for the noise model at hand turns the score into the estimate $\hat{x}$. From slide 29 of the course slides (Ye 2026c).

data with Gaussian noise have different marginals and need their own score networks (or one network conditioned on the noise). What the separation buys is that, for a given dataset, the noise model is needed only at inference, and the same training code applies to every noise type.

This decomposition — learn the score of a distribution during training, design the downstream use at inference — is precisely the structure of diffusion models. There, too, training only learns a score, and sampling is a separate algorithm built on top of it; the difference is that diffusion sampling iterates (denoise with Tweedie, then add a little less noise), while Noise2Score denoises deterministically in one step. What remains is the training phase: how do we estimate the score from noisy samples?

5.11. Estimating the score from noisy data

5.11.1. The score function

For a density $p_{data}(x)$ (slide 31), the score $\nabla_x \log p_{data}(x)$ is a vector field: at every point it points in the direction in which the density increases fastest, towards the nearby modes, with a magnitude that is large where the density changes quickly relative to its value (Figure 5.11). It is the same object that diffusion models learn. Two features make it convenient: it does not depend on the normalizing constant of p, since $\nabla \log(c\,p) = \nabla \log p$, and it can be estimated from samples. Before diffusion models, two classical estimators were known.

5.11.2. Denoising autoencoders estimate the score

The first is the denoising autoencoder (DAE). In the form shown on slide 32 (Alain and Bengio 2014), we take the *noisy* data y, corrupt it further with artificial Gaussian noise of level σ_a, and train the network to recover y:

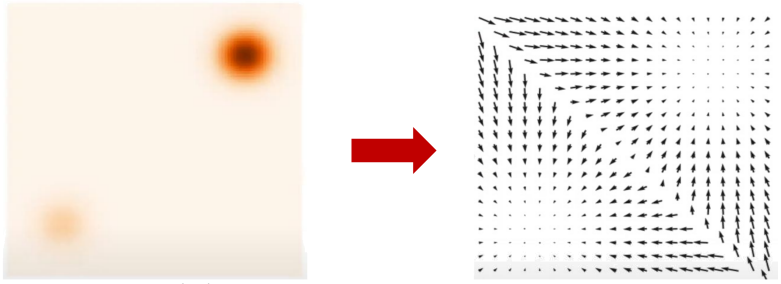


Figure 5.11.: A two-mode density (left) and its score (right). Each arrow points in the local direction of steepest increase of the log-density, and the arrows vanish at the modes. From slide 31 of the course slides (Ye 2026c); the slide credits Yang Song's blog post on score-based generative modelling.

$$\ell_{DAE}(\Theta) = \mathbb{E}_{\substack{y\sim P_Y \\ u\sim\mathcal{N}(0,I) \\ \sigma_a\sim\mathcal{N}(0,\delta^2)}} \left\| y - F_\Theta(y + \sigma_a u) \right\|^2. \tag{5.20}$$

Note that this is a Noise2X loss: the input is a modified version of y, the target is y.

Proposition 5.1 (The optimal DAE is a score estimator)**.** *Fix the artificial noise level σ_a and let $p_{\sigma_a} = p * \mathcal{N}(0, \sigma_a^2 I)$ be the density of $\tilde{y} = y + \sigma_a u$. The minimizer of Equation 5.20 at this level is*

$$F_{\Theta^*}(\tilde{y}) = \tilde{y} + \sigma_a^2 \nabla \log p_{\sigma_a}(\tilde{y}),$$

and as $\sigma_a \to 0$,

$$F_{\Theta^*}(y) = y + \sigma_a^2 \, l'(y) + o(\sigma_a^2). \tag{5.21}$$

Proof. By Lemma 5.1 the minimizer is $F^*(\tilde{y}) = \mathbb{E}[y \mid \tilde{y}]$. The pair $(y, \tilde{y})$ is exactly a "clean signal plus Gaussian noise" pair, with y in the role of the clean signal and σ_a in the role of σ, so the Gaussian Tweedie formula Equation 5.16 applies to it with the marginal density p_{σ_a} of $\tilde{y}$: $\mathbb{E}[y \mid \tilde{y}] = \tilde{y} + \sigma_a^2 \nabla \log p_{\sigma_a}(\tilde{y})$. For the limit, p_{σ_a} solves the heat equation, so $p_{\sigma_a} = p + \frac{\sigma_a^2}{2} \Delta p + o(\sigma_a^2)$ for smooth p, and therefore $\nabla \log p_{\sigma_a} = \nabla \log p + O(\sigma_a^2)$. Multiplying by σ_a^2 gives Equation 5.21. □

The score is then read off from the residual, $\hat{l}'(y) = (F_{\Theta^*}(y) - y)/\sigma_a^2$. Subtracting two nearly equal images and dividing by a tiny σ_a^2 is numerically fragile, so in practice the network is parameterized in residual form, $F_\Theta(\tilde{y}) = \tilde{y} + \sigma_a^2 R_\Theta(\tilde{y})$. For a *fixed* level σ_a, substituting $\tilde{y} = y + \sigma_a u$ into Equation 5.20 gives

$$\ell_{DAE}(\Theta) = \sigma_a^2 \, \mathbb{E} \left\| u + \sigma_a R_\Theta(y + \sigma_a u) \right\|^2, \qquad R_{\Theta^*}(\tilde{y}) = -\frac{\mathbb{E}[u \mid \tilde{y}]}{\sigma_a} = \nabla \log p_{\sigma_a}(\tilde{y}),$$

so R_Θ is trained directly as a score network. Three cautions keep the argument honest. First, at a fixed $\sigma_a > 0$ the DAE learns the score of the *smoothed* density p_{σ_a}, and only its rescaled residual converges to l' as $\sigma_a \to 0$; at $\sigma_a = 0$ the task is the plain autoencoder and the identity is optimal. Second, the slide samples $\sigma_a \sim \mathcal{N}(0, \delta^2)$; a network that is not told σ_a then learns a weighted mixture of smoothed scores, so the fixed-scale result holds for it only approximately. Third, the auxiliary level σ_a is used only to *learn* the score; denoising the real image uses the physical noise level, $\hat{x} = y + \sigma^2 R_{\Theta^*}(y)$, and confusing the two gives the wrong correction. Two further observations follow.

- **The limit is Tweedie for Gaussian noise.** Equation 5.21 has exactly the form $y + \sigma^2 l'(y)$ of Equation 5.16. The lecture asked the class where we had seen this formula before; this is the answer. A DAE trained at level σ_a outputs the Tweedie denoiser for Gaussian noise of level σ_a.
- **Noise2X methods are related to score estimation.** Noisier2Noise (Moran et al. 2020) adds further noise to the noisy image and regresses back to it, which is the DAE loss with y as target; Moran et al. then apply a correction that depends on the noise model (for additive noise of the same level, $2F(z) - z$) to reach the clean signal, and their method is not restricted to Gaussian noise. Noise2Noise has the same population minimizer as supervised training (Theorem 5.1), the MMSE denoiser, and for Gaussian noise that denoiser determines the score through Equation 5.17, so a perfectly trained Noise2Noise network *is* the Gaussian Tweedie denoiser. Blind-spot methods target the restricted conditional mean $\mathbb{E}[x_J \mid y_{J^c}]$ instead, which is related to, but in general not equal to, the Tweedie denoiser. This is the sense in which the lecture explains these methods through the score.

  As the lecture added, this reading is specific to Gaussian noise.

5.11.3. SURE is implicit score matching

The second classical estimator is score matching (Hyvarinen 2005) (slide 33). Plug the residual form

$$F_\Theta(y) = y + \sigma^2 R_\Theta(y), \qquad \hat{l}'(y) = \frac{F_{\Theta^*}(y) - y}{\sigma^2} = R_{\Theta^*}(y)$$

into the SURE loss Equation 5.9. (Slide 33 writes σ_a here; in SURE the level is the physical noise level σ, so we write σ.) The autoencoder term becomes $\|y - F_\Theta(y)\|^2 = \sigma^4 \|R_\Theta(y)\|^2$, and the divergence is $\operatorname{div}_y F_\Theta = \operatorname{div}_y y + \sigma^2 \operatorname{div}_y R_\Theta = n + \sigma^2 \operatorname{div}_y R_\Theta$. Hence

$$\begin{aligned}\ell_{SURE}(\Theta) &= \mathbb{E}_{y\sim P_Y}\{\|y - F_\Theta(y)\|^2 + 2\sigma^2 \operatorname{div}_y F_\Theta(y)\} \\ &= \mathbb{E}_{y\sim P_Y}\{\sigma^4 \|R_\Theta(y)\|^2 + 2\sigma^4 \operatorname{div}_y R_\Theta(y)\} + 2\sigma^2 n \\ &= 2\sigma^4\, \mathbb{E}_{y\sim P_Y}\left\{\tfrac{1}{2}\|R_\Theta(y)\|^2 + \operatorname{div}_y R_\Theta(y)\right\} + 2\sigma^2 n.\end{aligned}$$

Up to the positive factor $2\sigma^4$ and an additive constant, this is the **implicit score matching** loss of Hyvarinen (2005) with $\Psi_\Theta = R_\Theta$:

$$\ell_{ISM}(\Theta) = \mathbb{E}_{y\sim P_Y}\left\{\tfrac{1}{2}\|\Psi_\Theta(y)\|^2 + \operatorname{div}_y \Psi_\Theta(y)\right\}. \tag{5.22}$$

Theorem 5.5 (Implicit score matching estimates the score)**.** *Assume that $p > 0$ is differentiable, that Ψ_Θ is (weakly) differentiable, that $\mathbb{E}\|\Psi_\Theta(y)\|^2$, $\mathbb{E}|\operatorname{div}_y \Psi_\Theta(y)|$ and $\mathbb{E}\|\nabla_y \log p(y)\|^2$ are finite, and that the boundary terms of the coordinatewise integration by parts vanish (for example, $p(y)\Psi_\Theta(y) \to 0$ as $\|y\| \to \infty$). Then*

$$\ell_{ISM}(\Theta) = \tfrac{1}{2}\,\mathbb{E}_{y\sim P_Y}\|\Psi_\Theta(y) - \nabla_y \log p(y)\|^2 - \tfrac{1}{2}\,\mathbb{E}_{y\sim P_Y}\|\nabla_y \log p(y)\|^2,$$

so Equation 5.22 is minimized by $\Psi_\Theta = \nabla_y \log p = l'$.

Proof. Expand the square: $\frac{1}{2}\mathbb{E}\|\Psi - \nabla \log p\|^2 = \frac{1}{2}\mathbb{E}\|\Psi\|^2 - \mathbb{E}[\Psi^\top \nabla \log p] + \frac{1}{2}\mathbb{E}\|\nabla \log p\|^2$. For the middle term use $p\,\nabla \log p = \nabla p$ and integrate by parts in each coordinate; the boundary terms vanish by assumption:

$$\mathbb{E}[\Psi^\top \nabla \log p] = \int \Psi(y)^\top \nabla p(y)\, dy = -\int \operatorname{div} \Psi(y)\, p(y)\, dy = -\mathbb{E}[\operatorname{div} \Psi].$$

Substituting and rearranging gives the claim. □

The boundary condition is not a formality to be waved through. Several of the noise models in Table 5.2 live on a half-line — Poisson, Gamma, exponential — and on a one-sided or bounded support the coordinatewise integration by parts leaves a term at the finite end as well, which has to be checked rather than discarded; it vanishes when $p(y)\Psi_\Theta(y) \to 0$ there too. The decay-at-infinity example above covers only the unbounded case.

So the SURE network, written in residual form, is trained to output the score, and the denoised image it produces is $y + \sigma^2 l'(y)$, once more the Gaussian Tweedie formula (by

Theorem 5.5). The lecture's conclusion (slide 34): **Noise2X and SURE are score-based approaches.** For Gaussian noise, a DAE learns the score of a smoothed density, SURE in residual form learns the score of $p(y)$ itself, and Noise2Noise learns the MMSE denoiser that is equivalent to that score; blind-spot methods learn a restricted version of the same conditional mean.

For Gaussian noise the link is exact in three places: any MMSE denoiser determines the score through Equation 5.17, a DAE identifies the score of the smoothed density, and SURE in residual form *is* implicit score matching. (The Noise2Noise and Noise2Self identities themselves do not need Gaussian noise; Gaussianity is what makes the conversion to a score simple.) The lecture extended the analogy to masked token prediction in language models and to diffusion models: in its reading, much of the success of modern self-supervised learning comes from the same decomposition into score estimation at training time and a well-designed downstream task at inference time. For discrete tokens this is an analogy between prediction tasks rather than a theorem about continuous image scores.

5.11.4. Results

Because Noise2Score separates score estimation from the denoising formula, it can use the correct formula for each noise model. Slides 35–36 compare it with a noise-specific self-supervised baseline (SURE for Gaussian, its Poisson analogue PURE, Noise2Void for Gamma) and with Noise2Self. Figure 5.12 shows the first set of images and Table 5.3 collects the PSNR values of both slides.



Figure 5.12.: Noise2Score ("Ours") against the ground truth (GT), a noise-specific baseline (SURE, PURE or Noise2Void) and Noise2Self (N2S) for Gaussian noise ($\sigma = 50$), Poisson noise ($\zeta = 0.05$) and Gamma noise ($k = 100$). The number in each panel is its PSNR in dB; the insets magnify the boxed region. From slide 35 of the course slides (Ye 2026c), originally from Kim and Ye (2021).

Table 5.3.: PSNR (dB) of the example images on slides 35–36 (Kim and Ye 2021). These are selected single images, not benchmark averages.

Noise	Image	Noisy	SURE / PURE / N2V	Noise2Self	Noise2Score
Gaussian, $\sigma = 50$	1	15.26	27.13 (SURE)	28.14	28.71
Gaussian, $\sigma = 50$	2	15.12	26.59 (SURE)	27.60	28.18
Poisson, $\zeta = 0.05$	1	18.01	25.19 (PURE)	25.52	26.04
Poisson, $\zeta = 0.05$	2	17.27	23.10 (PURE)	23.65	24.45
Gamma, $k = 100$	1	26.06	30.72 (N2V)	30.23	32.91
Gamma, $k = 50$	2	24.89	25.72 (N2V)	26.82	29.47

The improvement over Noise2Self is about 0.6 dB for Gaussian noise and 0.5–0.8 dB for Poisson noise, and grows to about 2.7 dB for Gamma noise. These six images show the size of the gains; they do not by themselves establish the cause. Plausible contributors are that Noise2Score uses the noisy centre pixel that a blind-spot network discards, and that its reconstruction rule is matched to the noise model, but separating these would need ablations that the slides do not report. Since PSNR is $10\log_{10}(M^2/\mathrm{MSE})$ for a fixed peak M, a 2.68 dB gain means about $10^{0.268} \approx 1.85$ times lower mean squared error. The lecture also pointed out that fine structures survive better: what used to be a collection of heuristics becomes one principled pipeline.

5.12. Where the independence assumption fails

Every identity in this chapter is carried by one assumption, and it is worth seeing them in one place before looking at what happens when they fail.

Table 5.4.: What each result assumes. Every row is an independence, a zero-mean or a regularity condition; none of them is checkable from the noisy data alone.

Result	The assumption that carries the equality
Supervised squared error (Lemma 5.1)	the conditional expectation $\mathbb{E}[x \mid y]$ exists
Noise2Noise (Theorem 5.1)	the target noise has zero conditional mean, $\mathbb{E}[w_2 \mid x, y_1] = 0$
Noise2Self / Noise2Void (Theorem 5.2)	the hidden values are withheld from the network, and the noise is independent across the partition
SURE (Theorem 5.3)	additive Gaussian noise of known σ, and F_Θ weakly differentiable with integrable divergence
Tweedie's formula (Theorem 5.4)	the likelihood is a known exponential family, and the posterior normalization may be differentiated under the integral
DAE as score estimator (Proposition 5.1)	the perturbation is Gaussian, and the score is read in the limit $\sigma_a \to 0$
Implicit score matching (Theorem 5.5)	the boundary terms of the integration by parts vanish

Each fails in a recognizable way when it does not hold.

- **Noise2Noise** needs $\mathbb{E}[w_2 \mid x, y_1] = 0$. If the two copies share a noise component — fixed-pattern noise of a sensor, a common bias, a registration error between the shots — that component is predictable from y_1 and the network learns to

reproduce it; Section 5.15.1.2 shows the optimum sliding from 0.8 to the identity as the correlation grows.

- **Noise2Self and Noise2Void** need the noise to be independent across the partition. Real camera noise is often spatially correlated after demosaicing, compression or the camera's processing pipeline, and microscopy detectors can produce line or stripe artifacts. A blind-spot network then predicts the noise at a pixel from its correlated neighbours and leaves it in place. Structured masks that hide the whole correlated neighbourhood (Broaddus et al. 2020) are one remedy.
- **SURE** needs Gaussian noise with known σ; Stein's lemma is a Gaussian identity.
- **Tweedie's formula and Noise2Score** need a known likelihood. The formula in Table 5.2 is applied pixel by pixel, which presumes noise that is independent across pixels. For Gaussian noise with a *known* covariance Σ, the vector form $\hat{x} = y + \Sigma\,\nabla_y \log p(y)$ derived above still holds; for unknown or non-exponential-family noise, the formula has to be rederived or the noise model estimated.

5.13. Connections and further reading

5.13.0.1. From denoising to diffusion models

The lecture closed with the bridge to diffusion models (slides 37–38). Tweedie's formula holds at *every* noise level. A diffusion model adds controlled Gaussian noise of many levels to clean images, $x_t = a_t x_0 + b_t \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ ($a_t = 1$, $b_t = \sigma_t$ in the variance-exploding form; $a_t = \sqrt{\bar{\alpha}_t}$, $b_t = \sqrt{1 - \bar{\alpha}_t}$ in DDPM), and trains one network to estimate the score $\nabla \log p_t(x_t)$ of every noisy marginal. Applying Equation 5.16 to the clean signal $a_t x_0$ observed with noise level b_t,

$$\mathbb{E}[x_0 \mid x_t] = \frac{x_t + b_t^2\,\nabla_{x_t} \log p_t(x_t)}{a_t}, \qquad \mathbb{E}[\epsilon \mid x_t] = -b_t\,\nabla_{x_t} \log p_t(x_t),$$

so the familiar ϵ-prediction network of DDPM (Ho et al. 2020) is the same score, rescaled. Generation then becomes an iterative downstream task: estimate the clean image with Tweedie's formula, add back a little less noise, and repeat (Y. Song et al. 2021). A posterior mean is not a sample (Figure 5.13 (c) shows it falling between the modes), which is why the sampler needs the noise schedule and the stochastic or deterministic dynamics that later lectures specify. Self-supervised denoising is the one-step, deterministic special case of this machinery, and generalizing it beyond denoising is the topic of the next lecture.

5.13.0.2. Further reading

- Efron (2011) is a short, readable account of Tweedie's formula and of empirical Bayes, including Robbins' formula for counts.

- Batson and Royer (2019) develops $\mathcal{J}$-invariance beyond denoising (for example, for calibrating classical denoisers), and Broaddus et al. (2020) adapts the blind spot to spatially correlated noise.
- Moran et al. (2020) gives the Noisier2Noise correction for general additive and multiplicative noise.
- Vincent (2011) proves that denoising score matching and score matching are equivalent, which is the bridge from the DAE of slide 32 to the training objectives of Ho et al. (2020) and Y. Song et al. (2021).

5.14. Summary

- Denoising is the inverse problem with $\mathcal{A} = I$. Classical methods regularize (sparsity, TV, RED, which are MAP estimates) or exploit non-local self-similarity (BM3D). Supervised networks such as DnCNN learn the posterior mean $\mathbb{E}[x \mid y]$ but require clean targets.
- **Noise2Noise** trains on two independent noisy copies. Because the target noise is zero-mean given the input, the loss differs from the supervised loss by a constant (Theorem 5.1). The choice of loss selects the statistic: L_2 the mean, L_1 the median, L_0 the mode.
- **Noise2Self / Noise2Void** train on one noisy copy with a blind spot. $\mathcal{J}$-invariance excludes the identity, and independence of the noise across pixels makes the self-supervised loss equal to the true error plus a constant (Theorem 5.2).
- **SURE** trains on one noisy copy with a divergence penalty; Stein's lemma makes it an unbiased estimate of the true error (Theorem 5.3), and the divergence is estimated by Monte Carlo.
- **Tweedie's formula** expresses the posterior mean of the natural parameter of any exponential-family noise model through the score $l'(y)$ of the noisy data (Theorem 5.4). For Gaussian noise, $\mathbb{E}[x \mid y] = y + \sigma^2 l'(y)$: denoising *is* score estimation.
- **Noise2Score** learns the score of the observed data and applies the Tweedie formula matched to the noise model (Gaussian, Poisson, Gamma, …). For Gaussian noise, the population optimum of SURE in residual form is the score (for a network class that can represent it), that of Noise2Noise the equivalent MMSE denoiser, a DAE the score of a smoothed density, and blind-spot methods a restricted conditional mean; this is the lecture's unified explanation of why they work. Noise2Noise and Noise2Self themselves do not require Gaussian noise.
- Learning a score at training time and using it through Tweedie's formula at inference time is the core of diffusion models.

5.15. Advanced topics

The lecture states the results below; what follows works them through with numbers, and adds one example that points at the next chapter. None of it is needed to follow the argument above.

5.15.1. Worked examples

Unlike the PSNR values above, which are copied from the slides, the examples in this section are original to this note. Each is small enough to reproduce exactly, and each ends with a short, self-contained NumPy/SciPy snippet that prints the numbers quoted in the text, with the seeds used for the figures (the figures add only plotting calls).

5.15.1.1. Tweedie's formula on a Gaussian mixture

Let the clean signal be a two-component Gaussian mixture, $p(x) = 0.4\,\mathcal{N}(x; -2, 0.3^2) + 0.6\,\mathcal{N}(x; 1.5, 0.6^2)$, and observe $y = x + w$ with $\sigma = 0.8$. Convolving each component with the noise gives the noisy marginal in closed form,

$$p(y) = \sum_k \pi_k\,\mathcal{N}(y; \mu_k, s_k^2 + \sigma^2), \qquad l'(y) = \frac{\sum_k \pi_k\,\mathcal{N}(y; \mu_k, s_k^2 + \sigma^2)\,\frac{\mu_k - y}{s_k^2 + \sigma^2}}{\sum_k \pi_k\,\mathcal{N}(y; \mu_k, s_k^2 + \sigma^2)},$$

with (π_k, μ_k, s_k) the mixture weights, means and standard deviations. We draw 4×10^5 pairs (x, y), estimate $\mathbb{E}[x \mid y]$ by averaging x in narrow bins of y, and compare with $y + \sigma^2 l'(y)$ (Figure 5.13). Over the 45 bins of width 0.2 between -4.5 and 4.5 (bins with fewer than 50 samples skipped), the largest discrepancy is 0.025, which reflects Monte-Carlo error together with the bias of comparing a bin average with the value at the bin centre. The denoiser is strongly nonlinear: near each mode it shrinks y towards that mode, and between the modes it switches quickly from one to the other. This is exactly what the score does in panel (b), where it changes sign at each mode and between them.

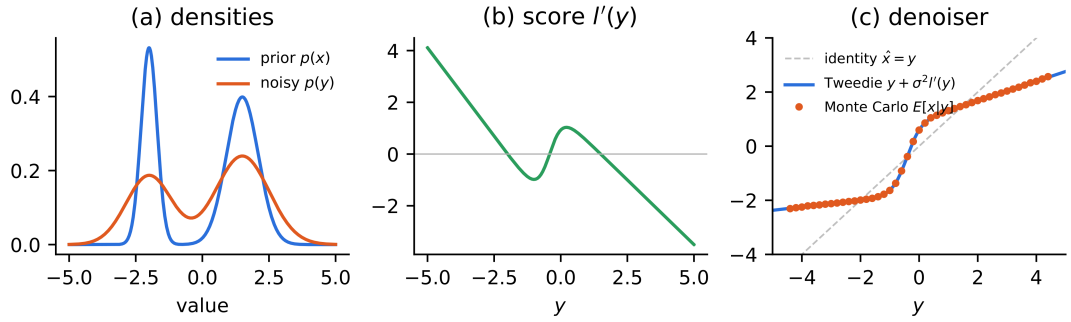


Figure 5.13.: Tweedie's formula on a 1D Gaussian mixture with $\sigma = 0.8$. (a) The clean prior and the noisy marginal $p(y)$. (b) The score $l'(y)$ of the noisy marginal. (c) The Tweedie denoiser $y + \sigma^2 l'(y)$ computed from the score alone, against the posterior mean estimated by Monte Carlo from clean-noisy pairs. The two coincide; the dashed line is the identity, i.e. no denoising.

The following code reproduces the numbers above; Figure 5.13 is plotted from the same arrays.

```
import numpy as np
from scipy.stats import norm

#-- 1. Tweedie on a Gaussian mixture (Figure: tweedie-1d)
rng = np.random.default_rng(1)
pi = np.array([0.4, 0.6]); mu = np.array([-2.0, 1.5])
s = np.array([0.3, 0.6]); sig = 0.8
v = s**2 + sig**2                          # variances of p(y)

def score(y):                              # l'(y) = d/dy log p(y)
    w = pi * norm.pdf(y[:, None], mu, np.sqrt(v))
    return (w * (mu - y[:, None]) / v).sum(1) / w.sum(1)

k = rng.choice(2, 400_000, p=pi)
x = rng.normal(mu[k], s[k])
y = x + sig * rng.normal(size=x.size)
edges = np.linspace(-4.5, 4.5, 46)         # bins of width 0.2
centre = 0.5 * (edges[1:] + edges[:-1])
b = np.digitize(y, edges)
mc = np.array([x[b == i].mean() if (b == i).sum() > 50 else np.nan
               for i in range(1, len(edges))])
tw = centre + sig**2 * score(centre)
print("1. max |Tweedie - MC| =", np.nanmax(np.abs(tw - mc)))
```

5.15.1.2. Three objectives, one optimum — and what correlation does

A scalar model shows Theorem 5.1 and Theorem 5.3 in closed form and also shows the failure mode of Noise2Noise quantitatively. Let $x \sim \mathcal{N}(0, 4)$, observe $y = x + w_1$ and a second copy $z = x + w_2$ with $w_1, w_2 \sim \mathcal{N}(0, 1)$ independent, and restrict the denoiser to $F_a(y) = ay$. Expanding each objective,

$$\begin{aligned} \text{supervised:} \quad & \mathbb{E}(ay - x)^2 = 4(a-1)^2 + a^2, \\ \text{Noise2Noise:} \quad & \mathbb{E}(ay - z)^2 = 4(a-1)^2 + a^2 + 1, \\ \text{SURE:} \quad & \mathbb{E}\big[(ay - y)^2 + 2a\big] - 1 = 5(a-1)^2 + 2a - 1 = 4(a-1)^2 + a^2, \end{aligned}$$

using $\mathbb{E}y^2 = 5$ and $\operatorname{div} F_a = a$. All three are minimized at $a^* = 4/5$, the Tweedie/Wiener coefficient $\tau^2/(\tau^2+\sigma^2)$. The Noise2Noise loss is larger by exactly $\mathbb{E}w_2^2 = 1$ (Equation 5.7), so comparing raw training-loss values across objectives is meaningless; only the minimizers agree.

Now let the two noises be correlated, $\mathbb{E}[w_1 w_2] = \rho$. The cross term of Theorem 5.1 no longer vanishes:

$$\mathbb{E}(ay - z)^2 = 4(a-1)^2 + a^2 + 1 - 2a\rho, \qquad a^*_\rho = \frac{4+\rho}{5}.$$

At $\rho = 0.6$ the optimum moves to 0.92, and at $\rho = 1$ (the two copies share their noise) it is $a = 1$, the identity: Noise2Noise then learns to copy its input. This is the independence condition of Theorem 5.1 failing in a measurable way.

The optima can be checked in one line each:

```
#-- 2. Scalar model: three objectives, correlated noise (exact)
for rho in (0.0, 0.6, 1.0):
    print("2. rho =", rho, " a* =", (4 + rho) / 5)
```

5.15.1.3. SURE without the clean image

Take a sparse vector $x \in \mathbb{R}^{2000}$ with 200 nonzero entries drawn from $\mathcal{N}(0, 4^2)$, observe $y = x + w$ with $\sigma = 1$, and denoise by soft thresholding, $F(y)_i = \operatorname{sign}(y_i)\max(|y_i| - \tau, 0)$. Its divergence is available in closed form, $\operatorname{div}_y F = \#\{i : |y_i| > \tau\}$, because each output coordinate has derivative 1 where it is not thresholded and 0 where it is. Figure 5.14 plots, as functions of τ, the true error (which uses x), SURE Equation 5.10 with the exact divergence, and SURE with the Monte-Carlo divergence Equation 5.11 using a single probe ($\epsilon = 10^{-3}$). On a grid of τ with step 0.05, the three curves track each other. The true optimum is $\tau = 1.25$ with error 0.254 per entry; SURE with the exact divergence picks the same $\tau = 1.25$, and the one-probe Monte-Carlo version picks $\tau = 1.10$, whose

true error (0.262) is within 0.01 of the optimum. SURE picks a near-optimal denoiser from the noisy data alone.

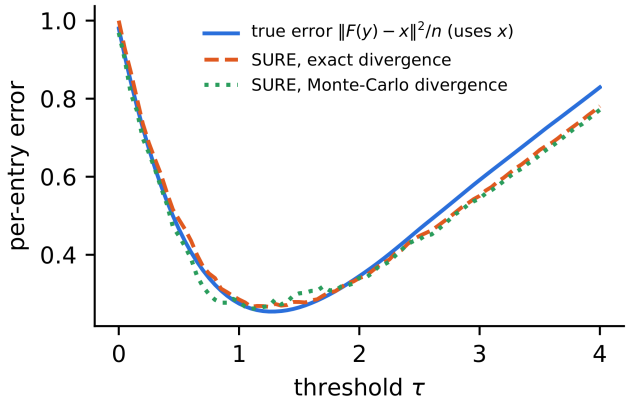


Figure 5.14.: SURE as a function of the soft-threshold τ for a sparse signal in Gaussian noise ($n = 2000$, $\sigma = 1$). The true per-entry error requires the clean signal; SURE, with the divergence computed exactly or by the one-probe Monte-Carlo estimate, requires only y and tracks the true error closely.

The code below produces the numbers in the text and the curves of Figure 5.14.

```
import numpy as np

#-- 3. SURE for soft thresholding (Figure: sure-check)
rng = np.random.default_rng(4)
d, sg = 2000, 1.0
xs = np.zeros(d)
xs[rng.choice(d, 200, replace=False)] = rng.normal(0, 4, 200)
ys = xs + sg * rng.normal(size=d)
probe, eps = rng.normal(size=d), 1e-3
soft = lambda u, t: np.sign(u) * np.maximum(np.abs(u) - t, 0)
taus = np.linspace(0, 4, 81)              # grid with step 0.05
true, sure, sure_mc = [], [], []
for t in taus:
    F = soft(ys, t)
    fit_term = np.sum((ys - F) ** 2) - d * sg**2
    div_mc = probe @ (soft(ys + eps * probe, t) - F) / eps
    true.append(np.sum((F - xs) ** 2) / d)
    sure.append((fit_term + 2 * sg**2 * np.sum(np.abs(ys) > t)) / d)
    sure_mc.append((fit_term + 2 * sg**2 * div_mc) / d)
for name, c in (("true", true), ("SURE", sure), ("MC-SURE", sure_mc)):
```

```
    i = int(np.argmin(c))
    print("3.", name, "tau* =", taus[i], " true error there =",
          round(true[i], 4))
```

5.15.1.4. How good is the Poisson formula?

With a Gamma prior on the Poisson rate everything is available in closed form, which lets us measure the error of Equation 5.18 exactly. Let $x \sim \text{Gamma}(a, b)$ (shape a, rate b) and $y \mid x \sim \text{Poisson}(x)$. The posterior is $\text{Gamma}(a + y, b + 1)$, so

$$\mathbb{E}[x \mid y] = \frac{a+y}{b+1}, \qquad \exp(\mathbb{E}[\log x \mid y]) = \frac{e^{\psi(a+y)}}{b+1}.$$

The marginal of y is negative binomial; extending it to real y through the Gamma function, $l(y) = \log\Gamma(a+y) - \log\Gamma(y+1) - \log\Gamma(a) + a\log\frac{b}{b+1} - y\log(b+1)$, so $l'(y) = \psi(a+y) - \psi(y+1) - \log(b+1)$. For $a = 4$, $b = 0.5$:

Table 5.5.: Posterior mean, posterior geometric mean, and the Poisson Tweedie formula for a Gamma(4, 0.5) prior.

y	$\mathbb{E}[x \mid y]$	$\exp(\mathbb{E}[\log x \mid y])$	$(y+\frac{1}{2})\exp(l'(y))$
0	2.667	2.341	2.085
1	3.333	3.006	2.955
2	4.000	3.672	3.648
5	6.000	5.670	5.662
10	9.333	9.002	8.999
20	16.000	15.668	15.666

Three things are visible in Table 5.5 and Figure 5.15. First, the formula is essentially exact for what it estimates, the geometric mean $\exp(\mathbb{E}[\log x \mid y])$, except at the smallest counts, where the approximation $\psi(y+1) \approx \log(y+\frac{1}{2})$ is crude. Second, the geometric mean sits below the arithmetic posterior mean by an almost constant amount, here about $1/(2(b+1)) = 0.33$, as predicted by $\psi(z) \approx \log z - \frac{1}{2z}$. Third, the posterior mean pulls the raw count towards the prior mean $a/b = 8$ (up for $y < 8$, down for $y > 8$); the other two columns are offset below it, so near $y = 8$ they sit slightly below both the count and the prior mean.

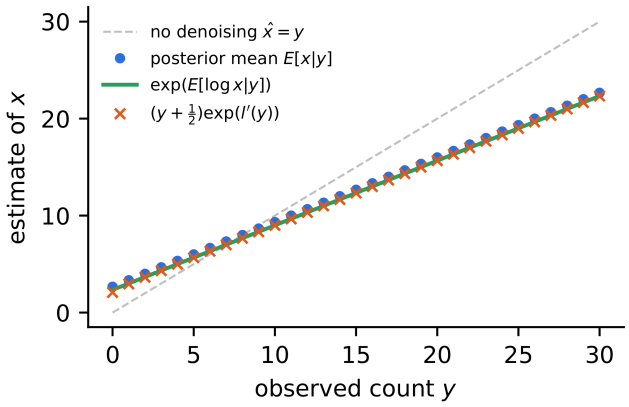


Figure 5.15.: Denoising Poisson counts with a Gamma(4, 0.5) prior: the exact posterior mean, the posterior geometric mean $\exp(\mathbb{E}[\log x \mid y])$, and the Tweedie formula $(y+\frac{1}{2})\exp(l'(y))$ computed from the score of the marginal. The formula tracks the geometric mean; both lie slightly below the posterior mean.

The code below produces Table 5.5.

```
import numpy as np
from scipy.special import digamma

#-- 4. Poisson Tweedie vs exact posterior, Gamma(4, 0.5) prior (exact)
a, bb = 4.0, 0.5
yy = np.arange(0, 31)
lp = digamma(a + yy) - digamma(yy + 1) - np.log(bb + 1)   # l'(y)
post_mean = (a + yy) / (bb + 1)
geo_mean = np.exp(digamma(a + yy)) / (bb + 1)
slide = (yy + 0.5) * np.exp(lp)
for j in (0, 1, 2, 5, 10, 20):
    print("4. y=%2d  %.3f  %.3f  %.3f"
          % (j, post_mean[j], geo_mean[j], slide[j]))
```

5.15.2. Tweedie's formula on a DDPM forward marginal

One application is worth writing out because the next chapter opens with it. A single step of the DDPM forward process produces

$$y = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{5.23}$$

with a mixing coefficient $\bar{\alpha}_t \in (0, 1]$, so that conditionally on x_0 the observation is Gaussian with mean $\sqrt{\bar{\alpha}_t}\, x_0$ and variance $1 - \bar{\alpha}_t$. Reading $\sqrt{\bar{\alpha}_t}\, x_0$ as the clean signal and $1 - \bar{\alpha}_t$ as the noise variance, the Gaussian case of Equation 5.16 gives

$$\sqrt{\bar{\alpha}_t}\, \mathbb{E}[x_0 \mid y] = y + (1 - \bar{\alpha}_t)\, \nabla_y \log p_t(y), \qquad \text{so} \qquad \mathbb{E}[x_0 \mid y] = \frac{y + (1 - \bar{\alpha}_t)\, \nabla_y \log p_t(y)}{\sqrt{\bar{\alpha}_t}}, \tag{5.24}$$

where p_t is the marginal density of the noisy variable at time t.

Nothing here is new: it is the identity of Equation 5.16 with the variance read off a particular forward process. What it shows is that the quantity a diffusion model spends its training budget estimating — the score of a noisy marginal — is the same quantity this chapter has been estimating, and that recovering the clean signal from it is a formula rather than a second network.

Based on the scribe note of Hyeonjae Kim, with material from Donggyu Kim, Mingeon Kim and Jeonghun Lee; edited by Jong Chul Ye.

6. Denoising Diffusion Probabilistic Models

6.1. Overview

A diffusion model splits the hard problem of sampling from a complex data distribution into two parts: an easy forward process that adds a little Gaussian noise many times, and a learned reverse process that undoes it. The forward process is fixed. It gradually erases the structure of the data until the state is close to a standard Gaussian. The reverse process starts from noise and is trained, as a neural network, to remove a little noise at every step (Ho et al. 2020).

This chapter develops that construction from first principles, following slides 2–25 of the lecture deck *Diffusion Model Theory* (Ye 2026e): the DDPM derivation on slides 2–16 and DDIM on slides 17–25. It begins with the probability tools needed to describe a latent Markov chain, derives the closed-form forward marginal and the variational training objective, and then turns the resulting Gaussian mean-matching problem into noise prediction. The relation among noise, data and velocity prediction makes the role of the loss weighting explicit. Finally, ancestral DDPM sampling is compared with the non-Markovian inference family and deterministic sampling rule of DDIM (Sohl-Dickstein et al. 2015; Ho et al. 2020; J. Song et al. 2021).

The construction rests on three mathematical facts.

1. Even after many Markov Gaussian transitions, x_t at any time can be sampled from x_0 in a single step, in closed form.
2. Fitting the intractable reverse conditionals becomes, through a variational bound, a sum of Gaussian KL divergences that can be computed.
3. Predicting the noise ϵ that the forward process added, instead of the reverse mean, turns each term of the objective into a weighted mean squared error, which DDPM then simplifies to a plain one.

The exact reverse conditional is generally not Gaussian, but it becomes locally close to Gaussian when each forward step is small. This motivates the many-step DDPM construction. DDIM instead changes the inference path while preserving the same noisy marginals, which permits deterministic and shortened sampling trajectories (Ho et al. 2020; J. Song et al. 2021). Short proofs follow the results they prove; longer verifications are collected in the appendix (Section 6.9).

After reading this note you should be able to:

- derive the closed form of $q(x_t \mid x_0)$ from $q(x_t \mid x_{t-1})$;
- decompose the variational bound on the negative log-likelihood into L_T, L_{t-1} and L_0;
- compute the mean and variance of $q(x_{t-1} \mid x_t, x_0)$ by completing the square;
- show that mean prediction and noise prediction describe the same reverse mean, and derive the noise-estimation loss;
- reproduce the DDPM training and ancestral sampling algorithms, and explain why sampling adds noise back;
- explain what it means that the non-Markovian forward process of DDIM has the same marginals as DDPM.

6.2. Notation

We follow the notation of the slides (Ye 2026e), which for DDPM is that of Ho et al. (2020), except that vectors are written without boldface: x_t, ϵ and μ_θ. References of the form "slide 12" point to that deck, so that each section can be checked against the lecture. The symbols are collected in Table 6.1.

Table 6.1.: Notation.

Symbol	Meaning
$x_0 \sim q_{\text{data}}$	clean sample from the data distribution, $x_0 \in \mathbb{R}^d$
x_t, $x_{a:b}$	latent at noise level t ($t = 1, \dots, T$); $x_{a:b} := (x_a, \dots, x_b)$
$\beta_t \in (0, 1)$	variance of the noise added at step t
$\alpha_t := 1 - \beta_t$	fraction of the signal kept in one step (in variance)
$\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, $\bar{\alpha}_0 := 1$	signal fraction accumulated from time 0 to t
$a_t := \sqrt{\bar{\alpha}_t}$, $b_t := \sqrt{1 - \bar{\alpha}_t}$	signal and noise coefficients in $x_t = a_t x_0 + b_t \epsilon$
$\mathrm{SNR}(t) := \bar{\alpha}_t / (1 - \bar{\alpha}_t)$	signal-to-noise ratio at time t
ϵ_t, $z \sim \mathcal{N}(0, I)$	noise of forward step t, and fresh noise of a sampling step; mutually independent
$\epsilon \sim \mathcal{N}(0, I)$, $\bar{\epsilon}_t$	noise of the one-step form $x_t = a_t x_0 + b_t \epsilon$ (written $\bar{\epsilon}_t$ when its time matters); independent of x_0, but along one path a function of $\epsilon_1, \dots, \epsilon_t$
q, q_t, q_γ	fixed forward diffusion, its marginal $q_t(x_t)$ at time t, and the DDIM family
p_θ	reverse generative model to be learned

Symbol	Meaning
$\tilde{\mu}_t, \tilde{\beta}_t$	mean and variance of the exact forward posterior $q(x_{t-1} \mid x_t, x_0)$
$\mu_\theta, \epsilon_\theta, \hat{x}_0$	reverse mean, noise prediction, and the data prediction computed from it
Σ_θ, s_θ	covariance of the reverse transition, and the score estimate $-\epsilon_\theta/\sqrt{1-\bar{\alpha}_t}$
σ_t^2	variance of the DDPM reverse transition
w_t	coefficient of the noise-prediction MSE at time t in the variational bound
γ_t^2 $(t \geq 2)$, γ_1^2	posterior variance of the DDIM family q_γ; $\gamma_1^2 > 0$ is the variance of its decoder $p_\theta(x_0 \mid x_1)$
η	interpolation between DDIM ($\eta = 0$) and DDPM ($\eta = 1$) through $\gamma_t = \eta\sqrt{\tilde{\beta}_t}$
v, ϕ_t	velocity target, and the angle with $a_t = \cos\phi_t$, $b_t = \sin\phi_t$
$\mathcal{N}(x; m, \Sigma)$	density at x of a Gaussian with mean m and covariance Σ

Norms are Euclidean unless stated otherwise. $\mathbb{E}_q$ denotes expectation over $x_0 \sim q_{\text{data}}$ and all latents generated by the forward diffusion.

6.3. Background: the Markov property and Gaussian tools

The derivation uses the KL divergence, Jensen's inequality and the evidence lower bound of a latent-variable model, which Section 1.6 and Section 4.3 develop. It also uses the Markov property and a few Gaussian identities, which this section states before they are applied.

6.3.1. Latent paths and the Markov factorization

The observed datum is a single x_0, but a diffusion model introduces latents $x_1, \dots, x_T$ at increasing noise levels, so its likelihood $p_\theta(x_0) = \int p_\theta(x_{0:T})\, dx_{1:T}$ is bounded as for the variational autoencoder of Chapter 4, with two differences: the latent is a whole diffusion path rather than a single code, and the "encoder" $q(x_{1:T} \mid x_0)$ is fixed rather than learned (Kingma and Welling 2014; Ho et al. 2020).

Definition 6.1 (First-order Markov chain)**.** A sequence $x_0, x_1, \dots, x_T$ is a first-order Markov chain if for all $t = 1, \dots, T$

$$q(x_t \mid x_{0:t-1}) = q(x_t \mid x_{t-1}). \tag{6.1}$$

Equivalently, $x_t \perp\!\!\!\perp x_{0:t-2} \mid x_{t-1}$.

Substituting Equation 6.1 into the chain rule gives $q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{0:t-1}) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$. This factorization defines the forward path distribution used below.

Proposition 6.1 (A Markov chain read backwards is still Markov)**.** *If the forward chain is Markov, then* $q(x_{t-1} \mid x_{t:T}) = q(x_{t-1} \mid x_t)$ *for all* t*. Proof in Section 6.9.2.*

The **Markov property** of the reverse chain holds whatever the step size. What needs small steps is the approximation that the reverse transition $q(x_{t-1} \mid x_t)$ has a **Gaussian functional form**, which is discussed with the reverse process.

6.3.2. Chapman–Kolmogorov composition and marginals

Integrating out intermediate states composes several transitions:

$$q(x_t \mid x_0) = \int q(x_t \mid x_{t-1}) q(x_{t-1} \mid x_0)\, dx_{t-1}. \tag{6.2}$$

The first factor can be written $q(x_t \mid x_{t-1})$ instead of $q(x_t \mid x_{t-1}, x_0)$ because of the Markov property. In DDPM both factors are linear Gaussian, so the integral is Gaussian too.

It is important to distinguish the **joint distribution of a path** from the **marginal at one time**. The product of Markov transitions fixes how the whole path is coupled, but knowing only $q(x_t \mid x_0)$ does not determine the coupling between different times. DDIM uses exactly this freedom: it keeps the same $q(x_t \mid x_0)$ as DDPM while using a non-Markovian path (Section 6.5) (J. Song et al. 2021).

6.3.3. Gaussian tools

Every closed form in this chapter follows from three lemmas. Their proofs are in Section 6.9.1.

Lemma 6.1 (Linear combinations of independent Gaussians)**.** *If* $u \sim \mathcal{N}(m_u, \Sigma_u)$ *and* $v \sim \mathcal{N}(m_v, \Sigma_v)$ *are independent, then for constant matrices* A, B *and a vector* c*,* $Au + Bv + c \sim \mathcal{N}(Am_u + Bm_v + c,\ A\Sigma_u A^{\mathsf{T}} + B\Sigma_v B^{\mathsf{T}})$.

Lemma 6.2 (Completing the square)**.** *If A is symmetric positive definite and $h \in \mathbb{R}^d$, then*

$$-\tfrac{1}{2}x^\mathsf{T}Ax + h^\mathsf{T}x = -\tfrac{1}{2}\left(x - A^{-1}h\right)^\mathsf{T} A \left(x - A^{-1}h\right) + \tfrac{1}{2}h^\mathsf{T}A^{-1}h.$$

Hence a density proportional to $\exp(-\frac{1}{2}x^\mathsf{T}Ax + h^\mathsf{T}x)$ is $\mathcal{N}(A^{-1}h,\, A^{-1})$: ***the quadratic coefficient is the precision, and the inverse precision times the linear coefficient is the mean.***

Lemma 6.3 (KL divergence between isotropic Gaussians)**.** *If $q = \mathcal{N}(m_1, s_1^2 I_d)$ and $p = \mathcal{N}(m_2, s_2^2 I_d)$, then*

$$D_{\mathrm{KL}}(q\|p) = \frac{1}{2}\left[d\,\frac{s_1^2}{s_2^2} + \frac{\|m_1 - m_2\|^2}{s_2^2} - d + d\log\frac{s_2^2}{s_1^2}\right]. \tag{6.3}$$

In particular, if $s_1 = s_2 = s$ then $D_{\mathrm{KL}}(q\|p) = \|m_1 - m_2\|^2/(2s^2)$.

6.3.4. Scope and assumptions

- $x_0 \in \mathbb{R}^d$ is continuous and q_{data} has a finite second moment; only the discretized decoder of Section 6.4.2.5 treats 8-bit data as discrete. Images are usually scaled to $[-1, 1]$ per channel (Ho et al. 2020, sec. 3.3).
- $T < \infty$, $0 < \beta_t < 1$, $\alpha_t = 1 - \beta_t$, and the noise schedule is fixed before training.
- The ϵ_t at different steps are $\mathcal{N}(0, I)$ samples independent of x_0 and of each other.
- Gaussian variances are positive. The degenerate endpoint $\tilde{\beta}_1 = 0$ at $t = 1$ is handled by L_0.
- That the network represents the ideal conditional expectation is an idealization used for interpretation.

6.4. Denoising diffusion probabilistic models

This section derives DDPM in six steps: the fixed forward process (Section 6.4.1), the reverse process and its variational bound (Section 6.4.2), the tractable forward posterior (Section 6.4.3), the change from mean prediction to noise prediction (Section 6.4.4), the three equivalent prediction targets (Section 6.4.5), and training and ancestral sampling (Section 6.4.6).

6.4.1. The forward process

Figure 6.1 summarizes the structure of DDPM. The solid arrows $p_\theta(x_{t-1} \mid x_t)$, pointing right, form the learned generative chain from the standard Gaussian x_T back to the data x_0; the dashed arrow $q(x_t \mid x_{t-1})$, pointing left, is the fixed forward transition from data to noise (Ho et al. 2020, fig. 2).

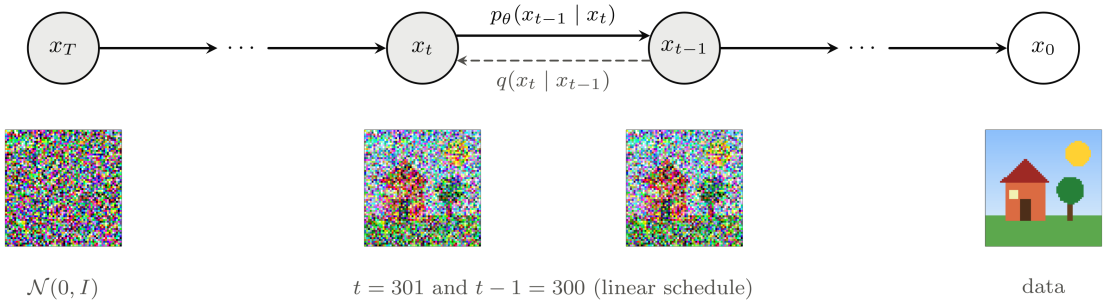


Figure 6.1.: A fixed forward chain and a learned reverse chain on the same latents. Solid arrows are the learned transitions $p_\theta(x_{t-1} \mid x_t)$ that turn noise into data; the dashed arrow is the fixed transition $q(x_t \mid x_{t-1})$ that adds noise. The thumbnails show a synthetic 64×64 image, drawn for this chapter from solid-colour shapes, as the data x_0, after $t - 1 = 300$ and $t = 301$ steps of the linear schedule, and as pure noise x_T.

The forward direction repeatedly attenuates the previous state and adds noise until the state is close to a standard Gaussian. The reverse direction models each denoising step as a transition with parameters μ_θ and Σ_θ; applying these transitions from x_T to x_0 produces a new sample rather than reconstructing a particular training example. Adding noise is fixed and tractable, whereas reversing the entropy increase requires information stored in the learned parameters. Small forward steps make a Gaussian approximation to each reverse transition plausible.

6.4.1.1. The forward chain

The forward process of DDPM (slide 4) is a Markov chain in the sense of Definition 6.1 with Gaussian transitions (Ho et al. 2020, Eq. 2)

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \qquad q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right) = \mathcal{N}(\sqrt{\alpha_t}\, x_{t-1}, \beta_t I)\,. \tag{6.4}$$

Reparameterizing the same transition gives

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, I). \tag{6.5}$$

The factor $\sqrt{\alpha_t}$ shrinks the previous signal and $\sqrt{\beta_t}$ adds fresh noise. Because $\alpha_t + \beta_t = 1$, this step preserves unit covariance, as the energy computation below makes precise. The $\{\beta_t\}_{t=1}^T$ are not learned but fixed in advance. This finite variance schedule determines both the forward corruption levels and the coefficients used by the reverse model.

6.4.1.2. Jumping to any time in one step

x_t can be sampled directly from x_0 without running the chain t times. Unrolling Equation 6.5 twice, as on slide 5, gives

$$\begin{aligned} x_t &= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{\beta_{t-1}}\epsilon_{t-1}\right) + \sqrt{\beta_t}\epsilon_t \\ &= \sqrt{\alpha_t\alpha_{t-1}}\, x_{t-2} + \underbrace{\sqrt{\alpha_t\beta_{t-1}}\,\epsilon_{t-1} + \sqrt{\beta_t}\,\epsilon_t}_{\sim\mathcal{N}(0,\,(1-\alpha_t\alpha_{t-1})I)}. \end{aligned}$$

The sum of two independent Gaussian noises is again Gaussian, with variance $\alpha_t\beta_{t-1}+\beta_t = \alpha_t(1-\alpha_{t-1})+(1-\alpha_t) = 1-\alpha_t\alpha_{t-1}$. Repeating this argument gives the following result.

Proposition 6.2 (Closed form of the forward marginal)**.** *For every $t = 0, 1, \dots, T$ there is $\bar{\epsilon}_t \sim \mathcal{N}(0, I)$, independent of x_0, such that*

$$\begin{aligned} x_t &= \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\bar{\epsilon}_t, \qquad \bar{\alpha}_t = \prod_{s=1}^{t}\alpha_s, \\ \textit{that is,}\quad q(x_t \mid x_0) &= \mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right). \end{aligned} \tag{6.6}$$

Proof. By induction on t; at $t = 0$, $\bar{\alpha}_0 = 1$ and the claim is trivial. Suppose that $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\bar{\epsilon}_{t-1}$, where $\bar{\epsilon}_{t-1}$ depends only on $\epsilon_1, \dots, \epsilon_{t-1}$. Substituting into Equation 6.5 and using $\alpha_t\bar{\alpha}_{t-1} = \bar{\alpha}_t$,

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \left(\sqrt{\alpha_t(1-\bar{\alpha}_{t-1})}\,\bar{\epsilon}_{t-1} + \sqrt{\beta_t}\,\epsilon_t\right).$$

The two noises in the parentheses are independent of each other and of x_0, so by Lemma 6.1 the parenthesis is Gaussian with mean 0 and covariance $[\alpha_t(1-\bar{\alpha}_{t-1}) + \beta_t]I = (1-\bar{\alpha}_t)I$, as in the two-step computation above. Writing it as $\sqrt{1-\bar{\alpha}_t}\,\bar{\epsilon}_t$ gives $\bar{\epsilon}_t \sim \mathcal{N}(0, I)$, independent of x_0. Equivalently, substituting the induction hypothesis into Equation 6.2 evaluates a Gaussian integral with the same result. □

This sample representation is the **one-step renoising formula**:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{6.7}$$

It matters a great deal for training: after drawing any t, x_t is built with one Gaussian draw, without generating the intermediate states $x_1, \dots, x_{t-1}$. As t grows and $\bar{\alpha}_t \to 0$, the data component disappears and $q(x_T \mid x_0) \approx \mathcal{N}(0, I)$, so generation can start from $x_T \sim \mathcal{N}(0, I)$. Multiplying the Markov transitions has therefore produced a direct transition from x_0 to any selected time t.

6.4.1.3. Energy, the noise schedule and the SNR

What "keeping the energy" means. Let Σ_{t-1} be the covariance of x_{t-1}. Since ϵ_t is independent of x_{t-1}, the covariance of the sum in Equation 6.5 is the sum of the two covariances; no Gaussian assumption on x_{t-1} is needed. Hence

$$\Sigma_t = \alpha_t \Sigma_{t-1} + \beta_t I \quad \Longrightarrow \quad \Sigma_t - I = \alpha_t(\Sigma_{t-1} - I) = \bar{\alpha}_t(\Sigma_0 - I).$$

If the data have unit covariance, the covariance stays I at every step (variance preserving); otherwise $\Sigma_t \to I$ as $\bar{\alpha}_t \to 0$. The mean is $\mathbb{E}x_t = \sqrt{\bar{\alpha}_t}\,\mathbb{E}x_0 \to 0$.

The linear schedule and the signal-to-noise ratio. The size of one step is set by β_t, but the signal left at time t is set by $\bar{\alpha}_t$, and the corresponding signal-to-noise ratio is $\mathrm{SNR}(t) = \bar{\alpha}_t/(1 - \bar{\alpha}_t)$. The original paper uses a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$ with $T = 1000$ (Ho et al. 2020, sec. 4). As Figure 6.2 shows, β_t is linear but $\bar{\alpha}_t$ decays nonlinearly: signal and noise variances become equal ($\bar{\alpha}_t = \frac{1}{2}$) between $t = 259$ and $t = 260$, and $\bar{\alpha}_T \approx 4.0 \times 10^{-5}$. The SNR spans more than eight orders of magnitude, from about 10^4 at $t = 1$ to 4×10^{-5} at $t = T$, and a uniform t covers them unevenly: the four decades above $\mathrm{SNR} = 1$ get only the first 259 steps, the remaining 4.4 decades the other 741.

Other schedules. Nichol and Dhariwal (2021, sec. 3.2) prescribe $\bar{\alpha}_t$ directly by the cosine schedule $\bar{\alpha}_t = f(t)/f(0)$, $f(t) = \cos^2\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)$ with $s = 0.008$, and set $\beta_t = 1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}$, clipped at 0.999. It destroys information more gradually: with $T = 1000$ its $\bar{\alpha}_t$ is closest to $\frac{1}{2}$ at $t = 496$, against $t = 259$ for the linear schedule (Section 6.6.4). A schedule is thus best seen as a design of the path of $\bar{\alpha}_t$, that is, of the SNR, rather than of β_t itself.

6.4.2. The reverse process and the variational bound

6.4.2.1. The generative model

The reverse generative model is

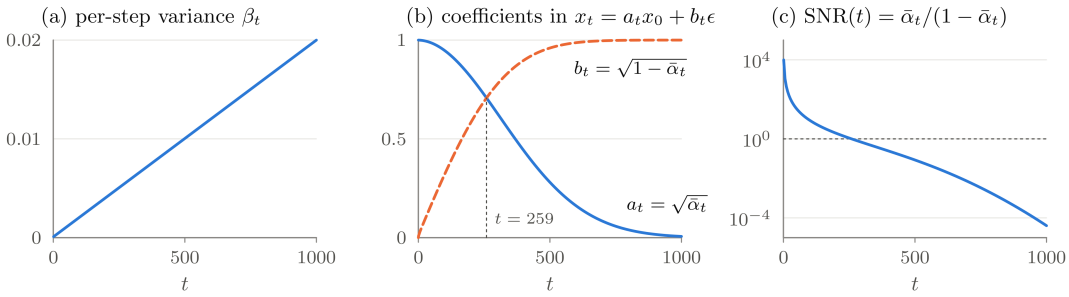


Figure 6.2.: A linear β_t removes the signal nonlinearly. (a) The per-step variance β_t of the original DDPM ($T = 1000$) is linear. (b) The coefficients $a_t = \sqrt{\bar{\alpha}_t}$ and $b_t = \sqrt{1-\bar{\alpha}_t}$ of the one-step renoising formula cross between $t = 259$ and $t = 260$ (dotted line at $t = 259$), so half of the signal variance is gone after about a quarter of the chain. (c) SNR(t) falls through more than eight orders of magnitude, from about 10^4 to 4×10^{-5}. Schedule of Ho et al. (2020), Sec. 4; curves from `curves()` in Section 6.6.4.

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \tag{6.8}$$

with $p(x_T) = \mathcal{N}(0, I)$ and, as in Ho et al. (2020), Eq. 1, and on slide 6, each reverse transition modelled as

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) .$$

A common choice fixes $\Sigma_\theta(x_t, t) = \sigma_t^2 I$ from the forward schedule and predicts only the mean. The time index is an explicit input because the noise level changes throughout the chain; Ho et al. (2020, sec. 4) feed t to a U-Net through a sinusoidal position embedding. The exact reverse conditional need not be Gaussian. Slide 6 states that in the limit of small steps it has the same functional form as the forward process; the next subsection explains why, and why large jumps require more care.

6.4.2.2. Why the reverse transition is nearly Gaussian for small steps

In Bayes' rule $q(x_{t-1} \mid x_t) \propto q(x_t \mid x_{t-1})\, q_{t-1}(x_{t-1})$, the first factor is a narrow Gaussian of width $\sqrt{\beta_t/\alpha_t}$. Expanding $\log q_{t-1}$ to first order inside that region and applying Lemma 6.2 gives the heuristic approximation (derived in Section 6.9.3)

$$q(x_{t-1} \mid x_t) \approx \mathcal{N}\left(\frac{x_t}{\sqrt{\alpha_t}} + \frac{\beta_t}{\alpha_t} \nabla \log q_{t-1}\!\left(\frac{x_t}{\sqrt{\alpha_t}}\right),\ \frac{\beta_t}{\alpha_t} I \right). \tag{6.9}$$

One reverse step undoes the forward shrinkage, moves along the score $\nabla \log q_{t-1}$ (the gradient of the log-density) of the marginal, and carries noise of the same size as the forward step. The approximation keeps the slope of $\log q_{t-1}$ across one step and drops its curvature. For sharply concentrated data the dropped term has relative size of the order of

$$\frac{\beta_t}{\alpha_t(1-\bar{\alpha}_{t-1})} = \frac{\mathrm{SNR}(t-1)}{\mathrm{SNR}(t)} - 1, \tag{6.10}$$

the relative drop of the SNR over one step. The error has two sources.

- *Gaussian data.* The exact reverse conditional is Gaussian, but the variance of Equation 6.9 is too large by a relative amount that is at most this ratio, and reaches it as the data variance goes to zero; the mean errs only at second order (Section 6.9.3).
- *Non-Gaussian data.* The exact reverse conditional is not Gaussian at all. If $\log q_{t-1}$ is smooth, with curvature bounded on the scale $\sqrt{\beta_t}$ of one step, this error comes from the dropped higher-order terms of the expansion and also vanishes as $\beta_t \to 0$.

Both errors are small when **the noise variance of one step is much smaller than the noise variance accumulated so far**. This is a scale rather than a sharp condition, since between well-separated modes the curvature of $\log q_{t-1}$ can be larger. For the linear schedule the ratio is largest at $t = 2$, about 1.2, and at $t = 1$ it is undefined, so the last step $x_1 \to x_0$ is the hardest to approximate by a Gaussian.

6.4.2.3. An upper bound on the negative log-likelihood

The goal is a small negative log-likelihood. Multiplying and dividing by the forward path distribution, as on slide 7, gives

$$p_\theta(x_0) = \int p_\theta(x_{0:T})\, dx_{1:T} = \int q(x_{1:T} \mid x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\, dx_{1:T} = \mathbb{E}_q\left[\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\right],$$

and by Jensen's inequality

$$\mathbb{E}[-\log p_\theta(x_0)] \le \mathbb{E}_q\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\right] =: \mathcal{L}_{\mathrm{VLB}}. \tag{6.11}$$

Minimizing $\mathcal{L}_{\mathrm{VLB}}$ therefore minimizes an upper bound on the negative log-likelihood. Substituting $p_\theta(x_{0:T}) = p_\theta(x_0)\, p_\theta(x_{1:T} \mid x_0)$ shows that for fixed x_0 the gap of the bound is $D_{\mathrm{KL}}(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0))$, so the bound is tight only when the forward path distribution equals the model's posterior over paths.

6.4.2.4. Decomposing the variational bound

Theorem 6.1 (Decomposition of the variational bound)**.** *For Equation 6.4 and Equation 6.8,*

$$\begin{aligned}\mathcal{L}_{\mathrm{VLB}} = &\underbrace{\mathbb{E}_q[D_{\mathrm{KL}}(q(x_T \mid x_0) \,\|\, p(x_T))]}_{L_T} \\ &+ \sum_{t=2}^{T} \underbrace{\mathbb{E}_q[D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))]}_{L_{t-1}} \\ &+ \underbrace{\mathbb{E}_q[-\log p_\theta(x_0 \mid x_1)]}_{L_0}.\end{aligned} \tag{6.12}$$

The algebra of slides 7–9 turns Equation 6.11 into Equation 6.12 in three steps; the full proof with every change of measure justified is in Section 6.9.4.

First, substituting Equation 6.4 and Equation 6.8, the loss is

$$\mathcal{L}_{\mathrm{VLB}} = \mathbb{E}_q\left[-\log p(x_T) - \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}\right]. \tag{6.13}$$

Next, the Markov property and Bayes' rule rewrite the denominator for $t \geq 2$:

$$q(x_t \mid x_{t-1}) = q(x_t \mid x_{t-1}, x_0) = q(x_{t-1} \mid x_t, x_0)\frac{q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)}. \tag{6.14}$$

Setting the $t = 1$ term aside and substituting this identity for $t \geq 2$, the terms $\log q(x_{t-1} \mid x_0) - \log q(x_t \mid x_0)$ telescope (slide 8):

$$\sum_{t\geq 1} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})} = \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} + \log p_\theta(x_0 \mid x_1) - \log q(x_T \mid x_0).$$

The telescoping sum is why all intermediate marginal-density terms disappear. Substituting it into Equation 6.13 and reading the expectations under q of the log-ratios as KL divergences gives Equation 6.12.

The roles of the terms are as follows. L_T measures how close the last forward distribution is to the prior and does not depend on θ. L_{t-1} matches one reverse step to the computable exact posterior; it is the main training signal of DDPM. The target $q(x_{t-1} \mid x_t, x_0)$ can be computed during training because x_0 is known. L_0 is the negative log-likelihood of the final reconstruction $x_1 \to x_0$. The Markov factorization and telescoping identity are what reduce a path-space expectation to this sum of local terms.

Figure 6.3 shows where on the chain each kind of term sits.

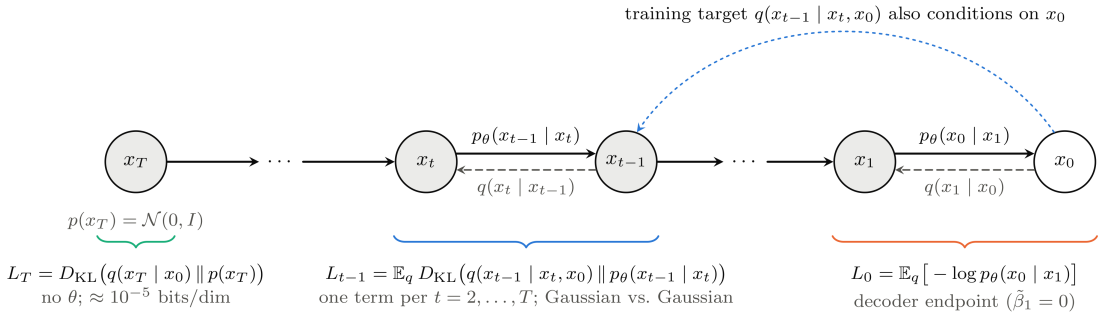


Figure 6.3.: Where each term of the bound sits on the chain. L_T compares the end of the forward chain with the prior and contains no parameters; each L_{t-1} is a Gaussian-to-Gaussian KL divergence at one interior transition; L_0 scores the final reconstruction in data space. Solid arrows are the learned transitions p_θ, grey dashed arrows the fixed transitions q, and the blue dotted arrow marks that the target $q(x_{t-1} \mid x_t, x_0)$ also conditions on the clean x_0.

6.4.2.5. The end terms of the bound

The prior term L_T. L_T has a closed form. Substituting $m_1 = \sqrt{\bar{\alpha}_T} x_0$, $s_1^2 = 1 - \bar{\alpha}_T$, $m_2 = 0$, $s_2^2 = 1$ into Lemma 6.3,

$$D_{\mathrm{KL}}(q(x_T \mid x_0) \,\|\, \mathcal{N}(0, I)) = \frac{1}{2}\left[\bar{\alpha}_T\left(\|x_0\|^2 - d\right) - d\log(1 - \bar{\alpha}_T)\right] \approx \frac{1}{2}\bar{\alpha}_T \|x_0\|^2.$$

Since $\|x_0\|^2 \le d$ for data scaled to $[-1, 1]^d$, the bracket is at most $-d\log(1 - \bar{\alpha}_T)$, so $L_T \le -\frac{d}{2}\log(1 - \bar{\alpha}_T)$. Per dimension, $L_T/d \le -\frac{1}{2}\log(1 - \bar{\alpha}_T) \approx 2.0 \times 10^{-5}$ nats $\approx 2.9 \times 10^{-5}$ bits, and Ho et al. (2020) report about 10^{-5} bits/dim on real data (Sec. 4). The cost of $q(x_T \mid x_0) \approx \mathcal{N}(0, I)$ is negligible.

The decoder term L_0. Ho et al. (2020) scale the 8-bit pixel values $\{0, \ldots, 255\}$ linearly to $[-1, 1]$, so neighbouring values are $2/255$ apart, and use the discrete decoder (Sec. 3.3, Eq. 13)

$$\begin{aligned} p_\theta(x_0 \mid x_1) &= \prod_{i=1}^{d} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2)\, dx, \\ \delta_+(x) &= \begin{cases} \infty, & x = 1, \\ x + \frac{1}{255}, & x < 1, \end{cases} \\ \delta_-(x) &= \begin{cases} -\infty, & x = -1, \\ x - \frac{1}{255}, & x > -1. \end{cases} \end{aligned} \tag{6.15}$$

where i indexes the coordinates. Each factor is the probability of the bin of width $2/255$ around the observed value, with the two edge bins extended to infinity, so L_0, and with it the whole bound, is a code length for discrete data; Equation 6.11 then bounds a discrete negative log-likelihood by the same Jensen argument. The decoder needs its own positive variance σ_1^2, because the choice $\sigma_t^2 = \tilde{\beta}_t$ adopted below gives $\tilde{\beta}_1 = 0$; for concreteness this chapter takes $\sigma_1^2 = \beta_1$. This is a convention of the chapter, not a value stated by Ho et al. (2020); any value of the same order, such as $\tilde{\beta}_2 \approx 5.5 \times 10^{-5}$, behaves the same way below.

6.4.3. The forward posterior

Computing L_{t-1} requires $q(x_{t-1} \mid x_t, x_0)$. As on slide 10, Bayes' rule and the Markov property give

$$\begin{aligned} q(x_{t-1} \mid x_t, x_0) &= \frac{q(x_t \mid x_{t-1})\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} \\ &= \frac{\mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, \beta_t I)\ \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}} x_0, (1-\bar{\alpha}_{t-1}) I)}{\mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) I)}. \end{aligned}$$

The numerator is a product of two Gaussians, the second of which is the one-step marginal Equation 6.6, and the denominator does not depend on x_{t-1}. Expanding the squared norms in the exponents and keeping only the terms that contain x_{t-1},

$$\log q(x_{t-1} \mid x_t, x_0) = -\frac{1}{2}\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\|x_{t-1}\|^2 + \left(\frac{\sqrt{\alpha_t}}{\beta_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} x_0\right)^{\mathsf{T}} x_{t-1} + \text{const},$$

and completing the square (Lemma 6.2) gives the following (slides 10–11).

Proposition 6.3 (Forward posterior)**.** *For $t \geq 2$, $q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\big(\tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\big)$ with*

$$\begin{aligned} \tilde{\mu}_t(x_t, x_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, x_0, \\ \tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t. \end{aligned} \tag{6.16}$$

Proof. By Lemma 6.2 the precision is the coefficient of $-\frac{1}{2}\|x_{t-1}\|^2$ above and the variance is its inverse. Putting the precision over a common denominator and using $\bar{\alpha}_t = \alpha_t \bar{\alpha}_{t-1}$ and $\alpha_t + \beta_t = 1$,

$$\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}} = \frac{\alpha_t(1-\bar{\alpha}_{t-1}) + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})} = \frac{1-\bar{\alpha}_t}{\beta_t(1-\bar{\alpha}_{t-1})} = \frac{1}{\tilde{\beta}_t}.$$

The mean is the variance $\tilde{\beta}_t$ times the linear coefficient, and multiplying out, $\tilde{\beta}_t\sqrt{\alpha_t}/\beta_t = \sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)$ and $\tilde{\beta}_t\sqrt{\bar{\alpha}_{t-1}}/(1-\bar{\alpha}_{t-1}) = \sqrt{\bar{\alpha}_{t-1}}\,\beta_t/(1-\bar{\alpha}_t)$. □

It is important that this distribution is conditioned on the clean x_0 as well as on x_t. During training x_0 is known, so the target can be computed; during generation it is not, so $p_\theta(x_{t-1} \mid x_t)$ has to approximate the resulting reverse conditional. DDIM will preserve the same noisy marginals while changing how the latent path depends on x_0.

6.4.3.1. Choice of the reverse variance

Slide 13 fixes the reverse variance to $\sigma_t^2 := \tilde{\beta}_t$, one of the two fixed choices of Ho et al. (2020, sec. 3.2); the other is $\sigma_t^2 = \beta_t$. The former is optimal when x_0 is a single point, the latter when $x_0 \sim \mathcal{N}(0, I)$, and the two give similar results. Neither is chosen freely at each step: the forward schedule $\{\beta_t\}$ determines both in closed form. Their ratio $\tilde{\beta}_t/\beta_t = (1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)$ is close to 1 except at small t (Nichol and Dhariwal 2021, sec. 3.1, Fig. 1), the same region where the Gaussian approximation of Section 6.4.2.2 is worst. At $t = 1$, $\tilde{\beta}_1 = 0$ means that the forward posterior is a point mass, not that a logarithm of 0 must be computed; this endpoint is handled by L_0, and the sampler sets $z = 0$ at the last step.

6.4.4. From mean prediction to noise prediction

6.4.4.1. Simplifying the Gaussian KL divergence

With the reverse variance fixed, L_{t-1} is a KL divergence between two Gaussians: one is the posterior computed from the real data, the other the parameterized model to be estimated by a neural network. Choosing different variances leaves an additional variance term; choosing them equal leaves only **the squared distance between the means** (Lemma 6.3). Hence (slide 12)

$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\right\|^2\right] + C_t, \tag{6.17}$$

where $C_t \geq 0$ is a constant independent of θ that vanishes when $\sigma_t^2 = \tilde{\beta}_t$ (Section 6.9.5).

6.4.4.2. Replacing the dependence on the data by the noise

The target $\tilde{\mu}_t$ in Equation 6.17 depends on x_0, which the network has to infer from x_t at generation time. As on slide 12, inverting the one-step renoising formula Equation 6.7 rewrites that dependence as one on the noise, so that the network can be asked to predict the noise instead: $x_0 = (x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon)/\sqrt{\bar{\alpha}_t}$. Substituting this expression into Equation 6.16,

$$\begin{aligned}\tilde{\mu}_t &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\,x_t + \beta_t\sqrt{\bar{\alpha}_{t-1}}\,\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon\right)}{1-\bar{\alpha}_t} \\ &= \frac{1}{\sqrt{\alpha_t}}\left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{1-\bar{\alpha}_t}\,x_t - \frac{\beta_t\sqrt{1-\bar{\alpha}_t}}{1-\bar{\alpha}_t}\,\epsilon\right) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right).\end{aligned} \tag{6.18}$$

The second line factors out $1/\sqrt{\alpha_t}$ using $\sqrt{\bar{\alpha}_{t-1}}/\sqrt{\bar{\alpha}_t} = 1/\sqrt{\alpha_t}$ and $\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1}) = (\alpha_t - \bar{\alpha}_t)/\sqrt{\alpha_t}$; the last equality holds because $\alpha_t + \beta_t = 1$ makes the numerator $1 - \bar{\alpha}_t$. The exact posterior mean is now expressed through x_t and the noise ϵ that produced it.

6.4.4.3. The noise parameterization and its loss

$\tilde{\mu}_t$ is a training target computed from the data, whereas the network estimates μ_θ. Following Equation 6.18, the network can be parameterized to predict the noise $\epsilon_\theta(x_t, t)$ instead of the mean (slide 13):

$$\mu_\theta(x_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right). \tag{6.19}$$

The difference of the two means is a constant multiple of the difference of the noises,

$$\tilde{\mu}_t - \mu_\theta = \frac{\beta_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\left(\epsilon_\theta(x_t, t) - \epsilon\right).$$

Substituting this relation into Equation 6.17 and replacing the expectation over x_t by one over (x_0, ϵ) through Equation 6.7 gives the loss of slide 13,

$$\begin{aligned}L_{t-1} &= w_t\,\mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2\right] + C_t, \\ w_t &:= \frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}.\end{aligned} \tag{6.20}$$

This trains the network to estimate the noise used to construct x_t at the selected noise level. The noise-prediction MSE is therefore not an arbitrary surrogate: it is the Gaussian reverse-mean matching objective written in observable training variables.

6.4.4.4. The simplified objective

In practice DDPM drops the time coefficient w_t of Equation 6.20 and trains on the following objective, the loss minimized by Algorithm 1 on slide 14 (Ho et al. 2020, Eq. 14):

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\substack{x_0 \sim q_{\text{data}}, \\ t \sim \text{Unif}\{1,\dots,T\}, \\ \epsilon \sim \mathcal{N}(0,I)}} \left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right], \quad x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon. \tag{6.21}$$

6.4.4.5. The simplified objective is a reweighted ELBO

Dropping w_t changes how training effort is distributed over noise levels. The next calculation makes explicit what changes, what stays fixed at the population optimum, and why the reweighting can improve samples.

What changes. Apart from the end terms L_T and L_0, discussed below, $\mathcal{L}_{\text{VLB}}$ is $\sum_{t\geq 2} w_t \, \mathbb{E}\|\epsilon - \epsilon_\theta\|^2$ plus a constant, and $\mathcal{L}_{\text{simple}}$ replaces every w_t by 1, up to the factor $1/T$ from averaging over t (its $t = 1$ term corresponds to L_0, as shown below). For the two fixed variance choices,

$$\begin{aligned} \sigma_t^2 = \tilde{\beta}_t : \quad w_t &= \frac{\beta_t^2(1-\bar{\alpha}_t)}{2\alpha_t(1-\bar{\alpha}_t)(1-\bar{\alpha}_{t-1})\beta_t} = \frac{\beta_t}{2\alpha_t(1-\bar{\alpha}_{t-1})}, \\ \sigma_t^2 = \beta_t : \quad w_t &= \frac{\beta_t}{2\alpha_t(1-\bar{\alpha}_t)}. \end{aligned} \tag{6.22}$$

The ratio $\beta_t/(\alpha_t(1-\bar{\alpha}_{t-1}))$ in the first expression is the ratio Equation 6.10 that set the heuristic error scale of the Gaussian approximation in Section 6.4.2.2. Written with the SNR,

$$w_t = \frac{1}{2}\left(\frac{\text{SNR}(t-1)}{\text{SNR}(t)} - 1\right) \qquad (\sigma_t^2 = \tilde{\beta}_t), \tag{6.23}$$

so the exact VLB weights the noise-prediction error at each step by the relative loss of SNR in that step. Table 6.2 quantifies this for the linear schedule. With $\sigma_t^2 = \tilde{\beta}_t$ the weight at $t = 2$ is about 59 times the mean, and for $t \leq 100$ it decays roughly like $1/t$ (Figure 6.4). The first 1% of the chain carries 18.4% of the total weight, against 0.9% under $\mathcal{L}_{\text{simple}}$. The exact VLB thus concentrates the training signal on easy denoising problems with almost no noise, and $\mathcal{L}_{\text{simple}}$ sharply lowers their share.

Table 6.2.: Weights of the interior terms $t = 2, \dots, T$ for the linear schedule ($T = 1000$); the $t = 1$ term is L_0, whose coefficient w_1 is undefined for $\sigma_t^2 = \tilde{\beta}_t$ because $\tilde{\beta}_1 = 0$. The first four columns give w_t divided by its mean; the last two give the share of the total weight carried by $t \le 10$ and by $t \le 100$. Computed by the code in Section 6.6.4.

σ_t^2 / loss	$t = 2$	$t = 10$	$t = 100$	$t = 500$	share of $t \le 10$	share of $t \le 100$
VLB, $\tilde{\beta}_t$	58.9	8.49	1.01	0.54	18.4%	39.2%
VLB, β_t	29.1	7.87	1.08	0.59	12.8%	34.1%
$\mathcal{L}_{\text{simple}}$	1	1	1	1	0.9%	9.9%

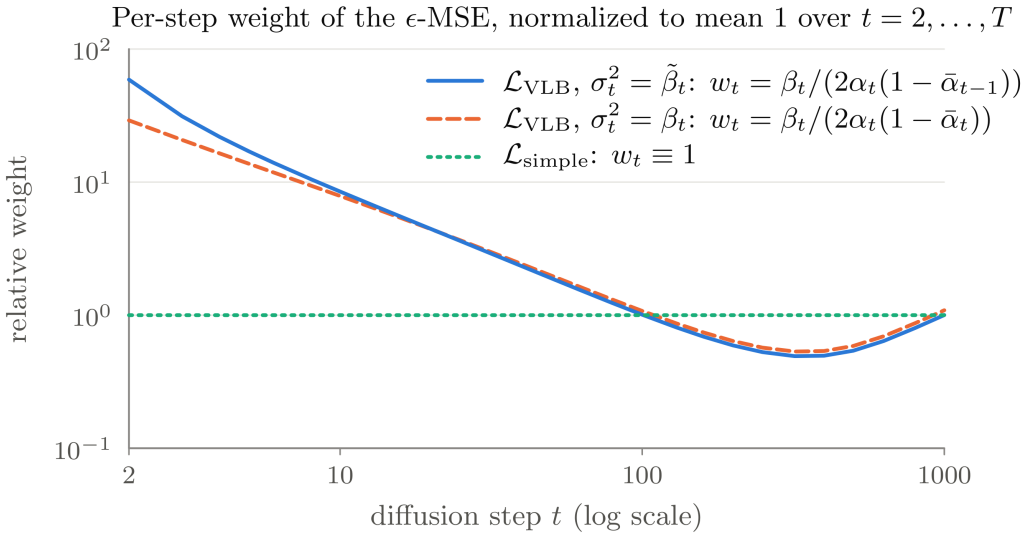


Figure 6.4.: The exact bound puts most of its weight on the first few steps. The time coefficients w_t of Equation 6.22 for both fixed variance choices, normalized to mean 1 over $t = 2, \dots, T$ (linear schedule, $T = 1000$, log scales), reach tens of times the average at small t and decay roughly like $1/t$; $\mathcal{L}_{\text{simple}}$ (dotted) weights all times equally. Values in Table 6.2.

What does not change. If the network can represent an arbitrary function of x_t for each t, then for any positive weights $\{\lambda_t\}$ the minimizer of $\sum_t \lambda_t \mathbb{E}\|\epsilon - \epsilon_\theta(x_t, t)\|^2$ is the same,

$$\epsilon^*(x_t, t) = \mathbb{E}[\epsilon \mid x_t] \tag{6.24}$$

(Section 6.9.6). The weights matter only in the practical situation where **one finite network is shared across all** t: they decide how capacity and gradient are allocated

across noise levels.

What it buys. Ho et al. (2020) explain that the simplified loss lowers the weight of the small-t terms, the easy tasks of removing very little noise, so that the network can focus on the harder denoising tasks at larger t, and that this improves sample quality (Sec. 3.4). On CIFAR-10, training on the exact VLB with a fixed isotropic variance gave IS 7.67, FID 13.51 and NLL $\leq$ 3.70 bits/dim, while $\mathcal{L}_{\text{simple}}$ gave IS 9.46, FID 3.17 and NLL $\leq$ 3.75 bits/dim (Ho et al. 2020, Tables 1–2). The reweighting **trades a little log-likelihood for a large gain in sample quality**. This benefit is empirical: the numerical examples of this chapter use the Bayes-optimal predictor, for which the weights do not matter (Equation 6.24), so they cannot show it; Nichol and Dhariwal (2021) recover much of the lost likelihood by also learning the variance.

The ablation of Ho et al. (2020, sec. 4.2, Table 2) shows why the noise parameterization matters: predicting $\tilde{\mu}_t$ works only with the true bound (FID 13.22 with a fixed isotropic variance; its unweighted MSE was unstable), whereas predicting ϵ is about as good under the bound (FID 13.51) and much better under $\mathcal{L}_{\text{simple}}$ (FID 3.17).

Learning the variance. The constant C_t of Equation 6.17 is independent of θ only because the variance is fixed. Nichol and Dhariwal (2021) instead learn a diagonal $\Sigma_\theta(x_t, t)$ with entries $\exp(\nu \log \beta_t + (1-\nu) \log \tilde{\beta}_t)$, an interpolation in the log domain between the two fixed choices, where the network outputs one ν per dimension (Sec. 3.1). The variance term C_t of the KL divergence then depends on θ and trains ν. They minimize $\mathcal{L}_{\text{hybrid}} = \mathcal{L}_{\text{simple}} + 0.001\, \mathcal{L}_{\text{VLB}}$ with a stop-gradient on the mean inside $\mathcal{L}_{\text{VLB}}$, so the mean keeps the reweighting of $\mathcal{L}_{\text{simple}}$ while the bound trains the variance. The rest of this chapter keeps the variance fixed, as on the slides.

Why not optimize the bound directly? Besides sample quality there is an optimization reason. The terms of the bound differ by orders of magnitude (Table 6.2), so with t drawn uniformly its gradient is very noisy. Nichol and Dhariwal (2021) keep the bound but sample t with probability proportional to $\sqrt{\mathbb{E}[L_{t-1}^2]}$ and divide each term by that probability, which leaves the bound unbiased and makes it far less noisy to optimize (Sec. 3.3).

The end terms. The weights w_t above apply only to the interior KL terms, $t \geq 2$; L_T does not depend on θ. If each bin integral of Equation 6.15 is approximated by density times bin width, $-\log p_\theta(x_0 \mid x_1)$ becomes $\|x_0 - \mu_\theta(x_1, 1)\|^2/(2\sigma_1^2)$ plus a constant. At $t = 1$, $x_1 = \sqrt{\alpha_1} x_0 + \sqrt{\beta_1}\epsilon$ and Equation 6.19 give $x_0 - \mu_\theta(x_1, 1) = \sqrt{\beta_1/\alpha_1}\,(\epsilon_\theta - \epsilon)$, so

$$L_0 \approx \frac{\beta_1}{2\alpha_1 \sigma_1^2}\, \mathbb{E}\|\epsilon - \epsilon_\theta(x_1, 1)\|^2 + \text{const},$$

formally a noise-prediction MSE with weight $1/(2\alpha_1) \approx \frac{1}{2}$ for $\sigma_1^2 = \beta_1$: up to this weight, the $t = 1$ term of $\mathcal{L}_{\text{simple}}$ (Ho et al. 2020, sec. 3.4). The approximation gives the form of

the term rather than its value, since the bin width $2/255 \approx 0.0078$ is not small next to $\sigma_1 = 0.01$.

Where the bound spends its nats. Figure 6.5 shows the small-t concentration from two sides: in (a), the two fixed variances $\tilde{\beta}_t$ and β_t differ only in the first steps; in (b), the interior terms L_{t-1}, computed exactly for the one-dimensional example of Section 6.6, fall by six orders of magnitude, as the weights of Table 6.2 predict. Nichol and Dhariwal (2021) report the same pattern for a trained image model (Fig. 2).

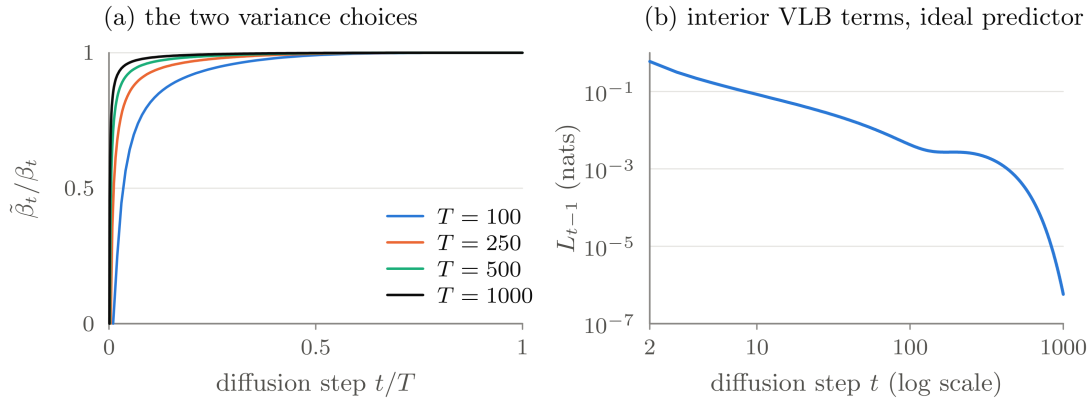


Figure 6.5.: The two variance choices differ only near $t = 0$, where the bound spends most of its nats. (a) The ratio $\tilde{\beta}_t/\beta_t$ for the linear schedule shortened to $T = 100, 250, 500, 1000$ steps (both endpoints of β_t multiplied by $1000/T$) is close to 1 except for the first steps. (b) The interior VLB terms L_{t-1} of the one-dimensional mixture of Section 6.6 with the Bayes-optimal predictor and $\sigma_t^2 = \tilde{\beta}_t$ (log axes) fall by six orders of magnitude, and $t \leq 10$ carries 42.6% of their total. Panel (a) from `curves()` in Section 6.6.4, panel (b) from `vlb_terms()` in the code of Section 6.6.

6.4.5. Noise, data and velocity prediction

Diffusion models can predict the noise ϵ, the clean data x_0, or the velocity v. These targets are algebraically interchangeable, but the corresponding MSE objectives weight reconstruction errors differently across noise levels (Salimans and Ho 2022, sec. 4 and App. D).

With $a_t = \sqrt{\bar{\alpha}_t}$ and $b_t = \sqrt{1 - \bar{\alpha}_t}$ we have $x_t = a_t x_0 + b_t \epsilon$ and $a_t^2 + b_t^2 = 1$.

6.4.5.1. Noise prediction and data prediction

If the network outputs $\hat{\epsilon} = \epsilon_\theta(x_t, t)$, the data prediction is

$$\hat{x}_0 = \frac{x_t - b_t \hat{\epsilon}}{a_t}, \tag{6.25}$$

and if it outputs $\hat{x}_0$ directly instead,

$$\hat{\epsilon} = \frac{x_t - a_t \hat{x}_0}{b_t}. \tag{6.26}$$

The two are algebraically equivalent at interior times with $a_t, b_t > 0$. But at very low SNR, $a_t \approx 0$ and the $1/a_t$ in Equation 6.25 strongly amplifies errors in the noise prediction; at very high SNR, $b_t \approx 0$ and the $1/b_t$ in Equation 6.26 strongly amplifies errors in the data prediction. Even though they represent the same ideal solution, the conditioning of training differs at finite precision and capacity. For the same reason samplers often clip the data prediction to the data range, $\hat{x}_0 \leftarrow \mathrm{clip}(\hat{x}_0, -1, 1)$, before it is used; this static thresholding keeps samples in range but departs from the exact noise-parameterized mean (Saharia et al. 2022, secs. 2.3, arXiv:2205.11487).

Whichever target is predicted, the DDPM reverse mean is the posterior mean of Equation 6.16 evaluated at the data prediction, $\mu_\theta(x_t, t) = \tilde{\mu}_t(x_t, \hat{x}_0)$: substituting Equation 6.25 for x_0 is exactly the computation of Equation 6.18 and gives back Equation 6.19. For the same underlying prediction the three parameterizations therefore define the same sampler; they differ only in what the network is asked to output.

6.4.5.2. Velocity prediction is an orthogonal rotation

Since $a_t^2 + b_t^2 = 1$, define the angle $\phi_t := \arctan(b_t/a_t) \in (0, \pi/2)$, so that $a_t = \cos\phi_t$ and $b_t = \sin\phi_t$. The velocity target is defined as

$$v := a_t \epsilon - b_t x_0 = \left.\frac{dx_t}{d\phi_t}\right|_{x_0, \epsilon \text{ fixed}}.$$

The second equality follows by differentiating $x_t = \cos\phi_t\, x_0 + \sin\phi_t\, \epsilon$ with respect to ϕ_t, which explains the name "velocity" (Salimans and Ho 2022, App. D). In matrix form,

$$\begin{bmatrix} x_t \\ v \end{bmatrix} = \begin{bmatrix} a_t & b_t \\ -b_t & a_t \end{bmatrix} \begin{bmatrix} x_0 \\ \epsilon \end{bmatrix},$$

the two rows are orthogonal and the determinant is $a_t^2 + b_t^2 = 1$, so the matrix is a rotation: (x_t, v) are the coordinates of the pair (x_0, ϵ) along axes turned by the angle ϕ_t. The inverse is the transpose,

$$x_0 = a_t x_t - b_t v, \qquad \epsilon = b_t x_t + a_t v. \tag{6.27}$$

For small t, $a_t \approx 1$ and $v \approx \epsilon$; for large t, $b_t \approx 1$ and $v \approx -x_0$. A v-predicting network enters the sampler through $\hat{x}_0 = a_t x_t - b_t \hat{v}$, that is, through $\tilde{\mu}_t(x_t, \hat{x}_0)$ as above.

This is why Salimans and Ho (2022) introduced v-prediction for progressive distillation (Sec. 4). A distilled sampler takes a few large steps and evaluates the network close to SNR $= 0$, where Equation 6.25 amplifies the error of $\hat{\epsilon}$ by $b_t/a_t \to \infty$, whereas $\hat{x}_0 = a_t x_t - b_t \hat{v}$ from Equation 6.27 stays bounded.

6.4.5.3. Same error, different weights

Converting one prediction into another at the same (x_t, t), Equation 6.25 and Equation 6.27 give $\epsilon - \hat{\epsilon} = \frac{a_t}{b_t}(\hat{x}_0 - x_0)$ and $x_0 - \hat{x}_0 = -b_t(v - \hat{v})$, hence

$$\|\epsilon - \hat{\epsilon}\|^2 = \frac{a_t^2}{b_t^2} \|x_0 - \hat{x}_0\|^2 = a_t^2 \|v - \hat{v}\|^2 . \tag{6.28}$$

Since $a_t^2/b_t^2 = \mathrm{SNR}(t)$ and $1/b_t^2 = 1 + \mathrm{SNR}(t)$,

$$\|\epsilon - \hat{\epsilon}\|^2 = \mathrm{SNR}(t)\, \|x_0 - \hat{x}_0\|^2, \qquad \|v - \hat{v}\|^2 = (1 + \mathrm{SNR}(t)) \|x_0 - \hat{x}_0\|^2.$$

Concretely, Algorithm 1 changes only in its last step. From the same draw (x_0, ϵ) and the same x_t, x_0-prediction minimizes $\|x_0 - \hat{x}_0(x_t, t)\|^2$, and v-prediction computes $v = a_t \epsilon - b_t x_0$ and minimizes $\|v - \hat{v}(x_t, t)\|^2$; either output is turned into $\hat{x}_0$ for the sampler by Equation 6.27. An unweighted ϵ-MSE is therefore an SNR-weighted error in x_0-space, and an unweighted v-MSE is an "SNR+1"-weighted one (Salimans and Ho 2022, sec. 4). As SNR$\to 0$ (large t), the ϵ-MSE puts almost no weight on the reconstruction error of x_0, while the v-MSE keeps a weight of at least 1. The three parameterizations are not different generative models but different coordinate systems for the same latent pair; they differ in **how much weight the error at each time receives** (Figure 6.6).

Back to the variational bound. The same conversion makes the VLB weights of Equation 6.23 transparent. Multiplying by SNR(t), which converts an ϵ-error into an x_0-error by Equation 6.28, each interior term is

$$L_{t-1} = \frac{1}{2}(\mathrm{SNR}(t-1) - \mathrm{SNR}(t))\, \mathbb{E}\|x_0 - \hat{x}_0(x_t, t)\|^2 \qquad (\sigma_t^2 = \tilde{\beta}_t), \tag{6.29}$$

which is the discrete-time loss of Kingma et al. (2021), Sec. 4, Eq. 13. The VLB weights the x_0-error at a step by **the SNR lost in that step**, whereas $\mathcal{L}_{\text{simple}}$ weights it by SNR(t) itself. In v-space, by the same identities, the VLB weight is $\frac{1}{2}(\mathrm{SNR}(t-1) - \mathrm{SNR}(t))/(1 + \mathrm{SNR}(t))$. The VLB weights sum to $\frac{1}{2}(\mathrm{SNR}(1) - \mathrm{SNR}(T))$, and since the linear schedule loses most of its SNR in the first steps (SNR$(1) \approx 10^4$), most of this total falls on small t.

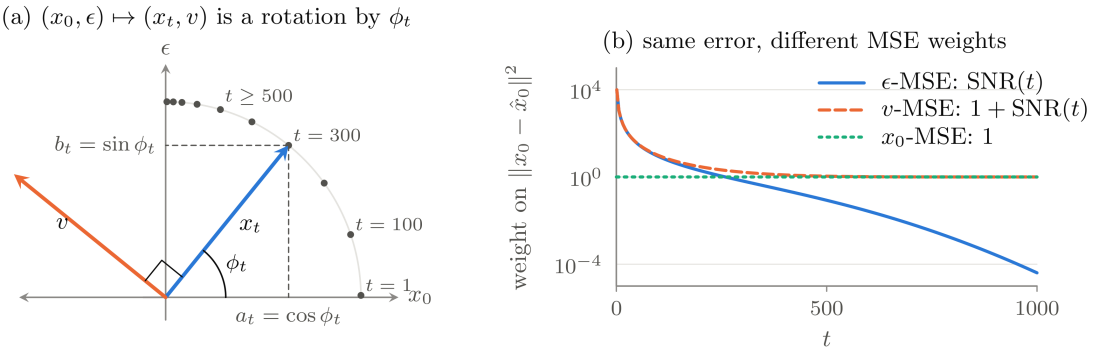


Figure 6.6.: Three prediction targets are one rotation with three loss weightings. (a) In the coefficient plane of (x_0, ϵ), the noisy input x_t and the velocity v are orthogonal axes turned by the angle ϕ_t; grey dots mark (a_t, b_t) along the linear schedule and are already bunched near the ϵ axis for $t \geq 500$. (b) The weights that the ϵ-, x_0- and v-MSE put on the same x_0 reconstruction error: SNR(t), 1 and $1+\text{SNR}(t)$ (log axis). Definitions and weights of Salimans and Ho (2022), Sec. 4 and App. D; SNR values from `curves()` in Section 6.6.4.

6.4.6. Training and ancestral sampling

6.4.6.1. The training algorithm

One training iteration of Equation 6.21 is (Ho et al. 2020, Algorithm 1; Ye 2026e, slide 14):

1. Draw $x_0 \sim q_{\text{data}}$.
2. Draw $t \sim \text{Unif}\{1, \dots, T\}$.
3. Draw the noise $\epsilon \sim \mathcal{N}(0, I)$, which is therefore known.
4. Build x_t in one step with Equation 6.7.
5. Update θ with $\nabla_\theta \|\epsilon - \epsilon_\theta(x_t, t)\|^2$.

Because the noise is drawn by us and added to clean data, the target ϵ is known exactly and no human labels are needed. Diffusion training is therefore **self-supervised**: the model constructs its own input-target pairs. Figure 6.7 shows the same clean image corrupted at several noise strengths, with the network estimating the added noise.

6.4.6.2. Reverse sampling

After training, draw $x_T \sim \mathcal{N}(0, I)$ and then $x_{t-1} \sim p_\theta(x_{t-1} \mid x_t)$ for each t in turn. This samples the joint $p_\theta(x_{0:T})$ in ancestral order; apart from the last step, which reports the mean of $p_\theta(x_0 \mid x_1)$ instead of a draw from it, the output is a sample of $p_\theta(x_0)$. With Equation 6.19 one step is (slides 13–14)

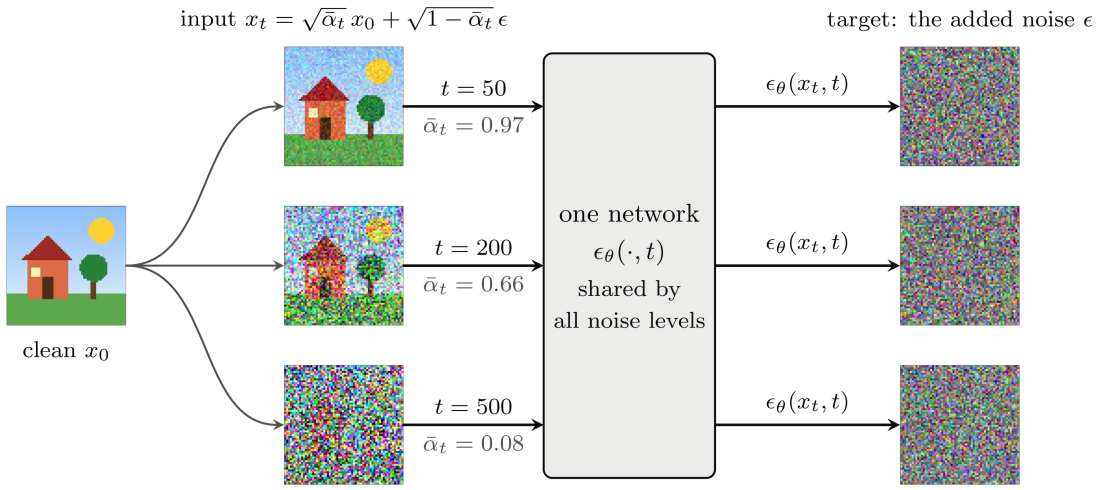


Figure 6.7.: Diffusion training makes its own labels. A synthetic clean image x_0, drawn for this chapter from solid-colour shapes, is noised with the one-step renoising formula at three levels of the linear schedule ($t = 50, 200, 500$, so $\bar{\alpha}_t = 0.97, 0.66, 0.08$), each with its own noise ϵ. A single network $\epsilon_\theta(\cdot, t)$, shared by all noise levels, receives x_t and t and must output the noise that was added.

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z, \qquad z \sim \mathcal{N}(0, I). \tag{6.30}$$

In full (Ho et al. 2020, Algorithm 2):

1. Draw $x_T \sim \mathcal{N}(0, I)$.
2. For $t = T, T-1, \ldots, 1$: draw $z \sim \mathcal{N}(0, I)$ if $t > 1$ and set $z = 0$ if $t = 1$, then compute x_{t-1} with Equation 6.30.
3. Return x_0.

The lecture puts the two algorithms side by side and highlights one line in each (Figure 6.8). In training it is the argument of ϵ_θ: the noisy image is built in a single jump with Equation 6.7, and no chain is ever simulated. In sampling it is the bracket: the model mean $\mu_\theta(x_t, t)$, to which the fresh noise $\sigma_t z$ is then added. Those two lines are the whole method; everything above derives them.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 6.8.: The two algorithms of Ho et al. (2020) as the lecture presents them (slide 14). Left, training: draw a clean image, a time and a noise vector, build x_t in one jump and take a gradient step on the squared error between ϵ and $\epsilon_\theta(x_t, t)$. Right, sampling: start from Gaussian noise and apply T reverse steps, each the model mean plus fresh noise of standard deviation σ_t. The boxed parts are the one-jump construction of x_t and the model mean. From slide 14 of the course slides (Ye 2026e), after Ho et al. (2020).

Because $1-\alpha_t = \beta_t$, the coefficient in Ho et al. (2020), Algorithm 2 is exactly $\beta_t/\sqrt{1-\bar{\alpha}_t}$. The final step sets $z = 0$: the output is $\mu_\theta(x_1, 1)$, the mean of the Gaussian inside the decoder, reported without noise (Ho et al. 2020, sec. 3.3). Training costs one network evaluation per sampled time, but ancestral sampling needs T sequential evaluations.

Written out, the two algorithms are shorter than their derivation. Each line below is one of the equations above: `x_t` is Equation 6.7, `loss` is Equation 6.21, and the reverse update is Equation 6.30 with $\sigma_t^2 = \tilde{\beta}_t$.

```
import torch

def ddpm_training_step(model, x_0, alpha_bar):
    """One training iteration (Algorithm 1)."""
    T = len(alpha_bar) - 1
    t = torch.randint(1, T + 1, (1,)).item()          # t ~ Unif{1,...,T}
    epsilon = torch.randn_like(x_0)                   # the target, known to us
    x_t = (torch.sqrt(alpha_bar[t]) * x_0             # one jump, no chain
           + torch.sqrt(1 - alpha_bar[t]) * epsilon)
    return torch.mean((epsilon - model(x_t, t)) ** 2)

@torch.no_grad()
def ddpm_sample(model, shape, alpha, alpha_bar, sigma):
    """Ancestral sampling (Algorithm 2)."""
    T = len(alpha) - 1
    x_t = torch.randn(shape)                          # x_T ~ N(0, I)
    for t in range(T, 0, -1):
        z = torch.randn_like(x_t) if t > 1 else torch.zeros_like(x_t)
        beta_t = 1 - alpha[t]
        x_t = (1 / torch.sqrt(alpha[t])
               * (x_t - beta_t / torch.sqrt(1 - alpha_bar[t]) * model(x_t, t))
               + sigma[t] * z)
    return x_t
```

Two details are worth reading off the code. Training never simulates the chain: one draw of t and one of ϵ give x_t directly, which is why a thousand-step model costs no more per iteration than a ten-step one. Sampling does the opposite, T sequential network evaluations with no way to parallelize over t, and that asymmetry is what the rest of the chapter, and much of the literature after it, works to remove.

Figure 6.9 shows the whole procedure on a two-dimensional example, in the spirit of the first figure of Sohl-Dickstein et al. (2015, sec. 2, Fig. 1). The forward chain spreads the data into noise, ancestral sampling carries the noise back, and the drift field shows how the reverse mean pulls samples towards the data: diffusion moves **the whole probability mass**, not a single blurred sample.

6.4.6.3. Why add noise back to the mean?

Why is $\sigma_t z$ added when ϵ_θ has already predicted a noise vector? The two quantities play different roles:

- $\epsilon_\theta(x_t, t)$ is the network output used to compute the conditional mean $\mu_\theta(x_t, t)$;

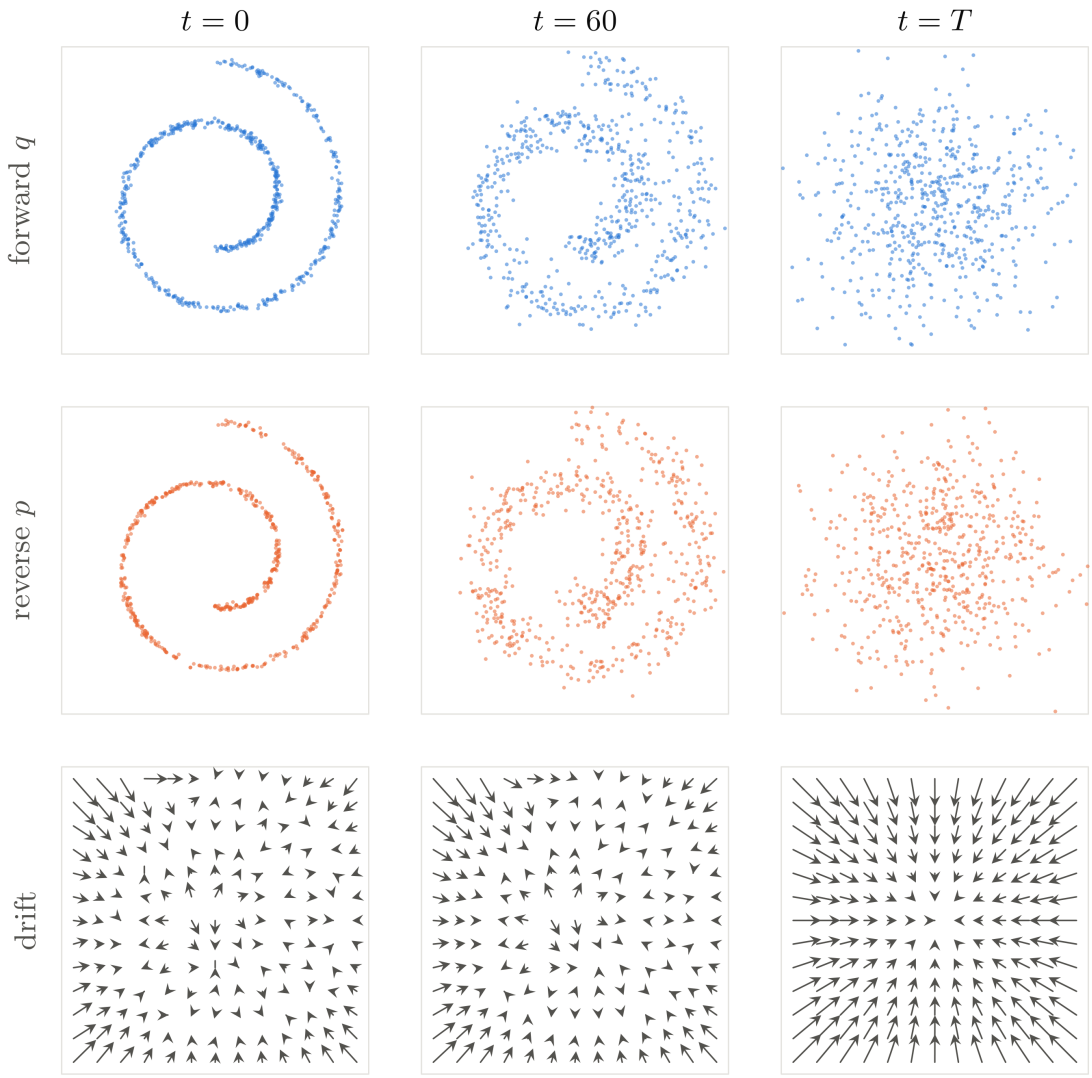


Figure 6.9.: Diffusion moves the whole probability mass, not a single sample. Top: forward samples of a Swiss roll at $t = 0$, 60 and T. Middle: 600 ancestral DDPM samples ($\sigma_t^2 = \tilde{\beta}_t$) recorded at the same times; generation reads this row from right to left. Bottom: the reverse drift $\mu(x_t, t) - x_t$ at $t = 1$, 60 and T, rescaled per panel. The reverse process uses the exact score of the empirical law of 1500 Swiss-roll points, so no network is trained, and the samples at $t = 0$ land on those training points: the figure shows transport of mass, not generalization. Values from `swiss_roll_figure()` (Section 6.6.3).

- z is a fresh random draw used to sample from the conditional Gaussian $p_\theta(x_{t-1} \mid x_t)$ centred at that mean.

DDPM minimizes a KL divergence between the posterior computed from data and a parameterized reverse Gaussian, so both sides are modelled as Gaussian transitions. Sampling from a Gaussian needs the mean plus noise of its standard deviation σ_t; using the mean alone picks a single point instead of sampling the conditional distribution. Without the injected noise the reverse chain can contract to almost a single point; the numerical example of Section 6.6 shows this. Removing the noise consistently also changes the mean, which is what DDIM does (Section 6.5).

6.4.6.4. The score, Tweedie's formula and the DDPM mean

Let $q_t(x_t) := \int q_{\text{data}}(x_0) q(x_t \mid x_0)\, dx_0$ be the marginal at time t. Under regularity that allows exchanging differentiation and integration (Section 6.9.7),

$$\nabla \log q_t(x_t) = -\frac{\mathbb{E}[\epsilon \mid x_t]}{\sqrt{1-\bar{\alpha}_t}} = -\frac{\epsilon^*(x_t, t)}{\sqrt{1-\bar{\alpha}_t}}, \qquad \mathbb{E}[x_0 \mid x_t] = \frac{x_t + (1-\bar{\alpha}_t)\nabla \log q_t(x_t)}{\sqrt{\bar{\alpha}_t}}. \tag{6.31}$$

The second identity is **Tweedie's formula**; plugging the optimal predictor into $\hat{x}_0$ of Equation 6.25 gives exactly this posterior mean. In the same way, writing the DDPM mean Equation 6.19 with the score $s_\theta := -\epsilon_\theta / \sqrt{1-\bar{\alpha}_t}$ gives

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t + \beta_t\, s_\theta(x_t, t)\Big), \tag{6.32}$$

which agrees with the mean of the small-step approximation Equation 6.9 to first order in β_t. Applying Tweedie's formula directly to the one-step model $x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{\beta_t}\epsilon_t$ gives

$$\mathbb{E}[x_{t-1} \mid x_t] = \frac{x_t + \beta_t \nabla \log q_t(x_t)}{\sqrt{\alpha_t}} \tag{6.33}$$

(Section 6.9.7), which is exactly Equation 6.32 with the true score. The ideal DDPM mean is therefore the posterior-mean (Tweedie) denoiser for one step, consistent with the minimizer of Equation 6.17 being $\mathbb{E}[\tilde{\mu}_t \mid x_t] = \mathbb{E}[x_{t-1} \mid x_t]$. Up to time weights, ϵ-prediction MSE is therefore denoising score matching (Vincent 2011), the denoising form of the score matching of Hyvarinen (2005).

6.4.6.5. Connection to Noise2Score

Tweedie's formula (Equation 6.31) connects noise prediction to the score, so one network that estimates the noise, equivalently the score of the noisy data, supplies the information needed for both denoising and generation. Noise2Score (Kim and Ye 2021) learns that score from noisy images alone and applies Tweedie's formula **once and deterministically** at the noise level of the image. DDPM sampling applies it **iteratively and stochastically** over a sequence of noise levels to create a new sample. Training one noise estimator and reusing it, repeatedly and for several downstream tasks such as denoising and generation, is what slide 14 calls the pattern of foundation models: pretraining followed by (iterative) downstream tasks.

6.5. DDIM: a non-Markovian forward process and deterministic sampling

DDIM, the subject of slides 17–25, changes the latent inference path while retaining the same family of noisy marginals (J. Song et al. 2021). Because those marginals determine the denoising inputs, the same noise-prediction network can be reused, under assumptions made precise below. This freedom yields a deterministic sampler and permits generation on a shorter subsequence of times. Slide 17 and the original DDIM paper write DDPM's $\bar{\alpha}_t$ as α_t; all equations below use the notation of this chapter (J. Song et al. 2021).

6.5.1. Motivation: is the Markov property needed?

DDPM specifies a Markov forward path and models its small reverse steps with Gaussian transitions. DDIM observes that the simplified loss Equation 6.21 depends on the forward process only through the noisy marginals $q(x_t \mid x_0)$, not through the coupling between different times. It therefore constructs a broader inference family in which a transition can depend directly on the clean endpoint x_0. The blue dashed arrows in Figure 6.10 (b) show this dependence. The generative update (Equation 6.36) still depends on the current state x_t and fresh noise only, so the generative process is Markov; the **forward inference distribution** q_γ is the non-Markovian object.

6.5.2. The non-Markovian forward family

For each sequence $\gamma = (\gamma_2, \dots, \gamma_T)$ with $0 \leq \gamma_t^2 \leq 1 - \bar{\alpha}_{t-1}$, define the family of slides 19–21 (J. Song et al. 2021, sec. 3.1, Eqs. 6–7)

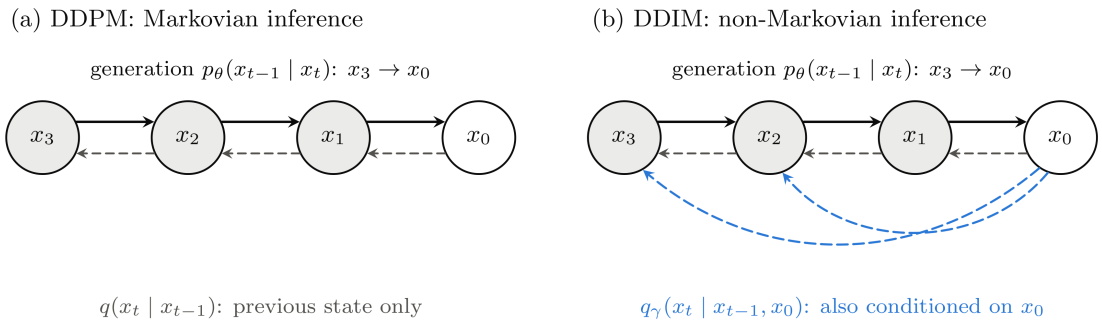


Figure 6.10.: DDIM keeps the marginals and changes only the inference path. (a) In DDPM each forward transition $q(x_t \mid x_{t-1})$ (grey dashed) depends only on the previous state. (b) In DDIM each forward transition $q_\gamma(x_t \mid x_{t-1}, x_0)$ also conditions on the clean x_0 (blue dashed). Both models share the generative chain $p_\theta(x_{t-1} \mid x_t)$ from x_3 to x_0 (solid) and the marginals $q(x_t \mid x_0)$.

$$q_\gamma(x_T \mid x_0) = \mathcal{N}\left(\sqrt{\bar\alpha_T}x_0, (1-\bar\alpha_T)I\right),$$
$$q_\gamma(x_{1:T} \mid x_0) = q_\gamma(x_T \mid x_0)\prod_{t=2}^{T} q_\gamma(x_{t-1} \mid x_t, x_0),$$

$$q_\gamma(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(\sqrt{\bar\alpha_{t-1}}\,x_0 + \sqrt{1-\bar\alpha_{t-1}-\gamma_t^2}\,\frac{x_t - \sqrt{\bar\alpha_t}x_0}{\sqrt{1-\bar\alpha_t}},\ \gamma_t^2 I\right). \tag{6.34}$$

The family is built so that the DDPM marginals survive.

Proposition 6.4 (Marginal preservation)**.** *The family Equation 6.34 has the same marginals as DDPM for every $t = 1, \dots, T$: $q_\gamma(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar\alpha_t}x_0, (1-\bar\alpha_t)I)$. Slide 20 states the result.*

Proof. By backward induction from $t = T$ (J. Song et al. 2021, App. B, Lemma 1). At $t = T$ the claim is the definition. If it holds at t, write $x_t = \sqrt{\bar\alpha_t}x_0 + \sqrt{1-\bar\alpha_t}\,\epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ and draw x_{t-1} from Equation 6.34 with a fresh $z \sim \mathcal{N}(0, I)$. Since $(x_t - \sqrt{\bar\alpha_t}x_0)/\sqrt{1-\bar\alpha_t} = \epsilon$,

$$x_{t-1} = \sqrt{\bar\alpha_{t-1}}x_0 + \sqrt{1-\bar\alpha_{t-1}-\gamma_t^2}\,\epsilon + \gamma_t z,$$

and since ϵ and z are independent, Lemma 6.1 gives the mean $\sqrt{\bar\alpha_{t-1}}x_0$ and the covariance $[(1-\bar\alpha_{t-1}-\gamma_t^2) + \gamma_t^2]I = (1-\bar\alpha_{t-1})I$. □

The coefficient $\sqrt{1-\bar{\alpha}_{t-1}-\gamma_t^2}$ is essential: with $\bar{\alpha}_t$ in place of $\bar{\alpha}_{t-1}$ the covariance would be $(1-\bar{\alpha}_t)I$ and the induction would break.

With the marginals in hand, the forward-time conditional follows from Bayes' rule (slide 19):

$$q_\gamma(x_t \mid x_{t-1}, x_0) = q_\gamma(x_{t-1} \mid x_t, x_0)\,\frac{q_\gamma(x_t \mid x_0)}{q_\gamma(x_{t-1} \mid x_0)}.$$

All three factors are Gaussian, so this conditional can be written out. Given x_0, write $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$ and draw x_{t-1} from Equation 6.34 as $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\gamma_t^2}\,\epsilon + \gamma_t z$. The pair is jointly Gaussian with $\mathrm{Cov}(x_t, x_{t-1} \mid x_0) = \sqrt{(1-\bar{\alpha}_t)(1-\bar{\alpha}_{t-1}-\gamma_t^2)}\,I$ and $\mathrm{Cov}(x_{t-1} \mid x_0) = (1-\bar{\alpha}_{t-1})I$, and conditioning on x_{t-1} gives, for $\gamma_t > 0$,

$$q_\gamma(x_t \mid x_{t-1}, x_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\,x_0 + \frac{\sqrt{(1-\bar{\alpha}_t)(1-\bar{\alpha}_{t-1}-\gamma_t^2)}}{1-\bar{\alpha}_{t-1}}(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\,x_0),\ \frac{(1-\bar{\alpha}_t)\,\gamma_t^2}{1-\bar{\alpha}_{t-1}}\,I\right). \tag{6.35}$$

For $\gamma_t^2 = \tilde{\beta}_t$ the coefficient of x_{t-1} is $\sqrt{\alpha_t}$, the x_0 terms cancel because $\sqrt{\bar{\alpha}_t} = \sqrt{\alpha_t}\sqrt{\bar{\alpha}_{t-1}}$, and the variance is β_t, so Equation 6.35 reduces to the DDPM transition Equation 6.4. For any other γ_t this transition depends on the clean image x_0 as well as on the previous state, so the inference path is not Markov, although its marginals are those of DDPM.

Because the marginals are the same, DDIM uses the same one-step renoising inputs as Equation 6.7. The relation between its variational objective and the simplified loss is derived below. The DDIM posterior mean should not be identified with the DDPM mean $\tilde{\mu}_t$ in general; the two coincide only for the variance choice $\gamma_t^2 = \tilde{\beta}_t$ proved in Proposition 6.5.

6.5.3. The generative step

At generation time x_0 is unknown, so the prediction $\hat{x}_0 = (x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t))/\sqrt{\bar{\alpha}_t}$ of Equation 6.25 is substituted for x_0 in Equation 6.34 (slides 21–22). Then $(x_t - \sqrt{\bar{\alpha}_t}\hat{x}_0)/\sqrt{1-\bar{\alpha}_t} = \epsilon_\theta(x_t, t)$ exactly, so one step is

$$x_{t-1} = \underbrace{\sqrt{\bar{\alpha}_{t-1}}\,\frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}}_{\text{predicted } x_0} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\gamma_t^2}\,\epsilon_\theta(x_t, t)}_{\text{direction pointing to } x_t} + \underbrace{\gamma_t z}_{\text{random noise}}. \tag{6.36}$$

This is the update of slide 17, J. Song et al. (2021), Sec. 4.1, Eq. 12, in the notation of this chapter. Comparing it with Equation 6.30 shows that both the fresh-noise coefficient and the deterministic mean direction depend on γ_t.

6.5.4. The exact relation to DDPM and deterministic sampling

Writing both samplers in the same coordinates makes their relation exact.

Proposition 6.5 (Reduction to DDPM)**.** *With $\gamma_t^2 = \tilde{\beta}_t$, Equation 6.34 equals the DDPM posterior $\mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t I)$. The DDPM ancestral step Equation 6.30 ($\sigma_t^2 = \tilde{\beta}_t$) can therefore be written*

$$x_{t-1}^{\text{DDPM}} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\,\epsilon_\theta(x_t,t) + \sqrt{\tilde{\beta}_t}\,z. \tag{6.37}$$

Proof in Section 6.9.9.

Deterministic DDIM ($\gamma_t = 0$), on the other hand, is

$$x_{t-1}^{\text{DDIM}} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\epsilon_\theta(x_t,t). \tag{6.38}$$

The two differ in exactly two places. (i) The variance $\tilde{\beta}_t$ that DDPM injects as fresh noise is carried deterministically by DDIM along the predicted noise direction ϵ_θ, so the coefficient of the "direction pointing to x_t" grows from $\sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}$ to $\sqrt{1-\bar{\alpha}_{t-1}}$. (ii) There is no z term. Seen in the original DDPM form Equation 6.30, this is why the mean looks different and there seems to be an extra ϵ_θ term.

Intermediate choices are parameterized as $\gamma_t = \eta\sqrt{\tilde{\beta}_t}$, with $\eta = 1$ giving DDPM and $\eta = 0$ giving DDIM (J. Song et al. 2021, sec. 5, Eq. 16). With $\gamma_t = 0$ the whole path down to x_0 is fixed as soon as x_T is fixed, so DDIM is an implicit probabilistic model; the initial x_T is still drawn at random.

6.5.4.1. Denoise, then re-noise

The three terms of Equation 6.36 were named after J. Song et al. (2021), which is how they are usually quoted. The lecture renames them, and the renaming is the point of the slide (slide 23):

$$x_{t-1} = \underbrace{\sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0(x_t,t)}_{\text{Tweedie's denoiser}} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\gamma_t^2}\,\epsilon_\theta(x_t,t)}_{\text{parametric noise}} + \underbrace{\gamma_t\, z}_{\text{stochastic noise}}. \tag{6.39}$$

The first term is not merely a prediction of x_0: by Equation 6.31 it is the Tweedie denoiser of Section 6.4.6.5 applied at noise level t — the same Theorem 5.4 that the previous chapter derived for self-supervised denoising. What the second and third terms do is put the sample back at the noise level $t-1$, splitting that re-noising between a direction the network computes and a draw that it does not. Every sampler in this chapter is one denoising step followed by one re-noising step, and η decides only how much of the re-noising is random.

This also explains why the noise cannot simply be dropped from DDPM while the mean is kept. The marginal of x_{t-1} has to carry variance $1-\bar{\alpha}_{t-1}$; take away the $\gamma_t z$ term and the coefficient of ϵ_θ must grow to make up the difference, which is exactly Equation 6.38. The lecture draws this as a sequence of manifolds $\mathcal{M}_t$, one per noise level, with each step dropping to the data manifold and climbing back to the next one up (slide 24). Keeping that transition on the right manifold is what the later chapters return to.

6.5.5. Why the same network serves every variance choice

The common noise-prediction objective follows from Theorem 1 of J. Song et al. (2021, sec. 3.2). Because $q_\gamma(x_{1:T} \mid x_0)$ is defined directly in reverse order, its variational bound splits into KL terms like Equation 6.12 without the Bayes rearrangement Equation 6.14. In each interior term both Gaussians have covariance $\gamma_t^2 I$, and their means differ only through x_0 versus $\hat{x}_0$. Let c_t be the coefficient of x_0 in the mean of Equation 6.34,

$$c_t := \sqrt{\bar{\alpha}_{t-1}} - \sqrt{1-\bar{\alpha}_{t-1}-\gamma_t^2}\,\sqrt{\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}} > 0.$$

Then each interior term is a weighted noise-prediction error,

$$\mathbb{E}_q\big[D_{\mathrm{KL}}\big(q_\gamma(x_{t-1} \mid x_t, x_0)\,\|\,p_\theta(x_{t-1} \mid x_t)\big)\big] = \lambda_t\,\mathbb{E}\|\epsilon - \epsilon_\theta(x_t,t)\|^2, \qquad \lambda_t = \frac{c_t^2}{2\gamma_t^2\,\mathrm{SNR}(t)}, \tag{6.40}$$

for every $\gamma_t > 0$ (Section 6.9.8). Here $p_\theta(x_{t-1} \mid x_t)$ is the generative step (Equation 6.36), and by Proposition 6.4 the expectation runs over the same pairs $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ as $\mathcal{L}_{\text{simple}}$. For $\gamma_t^2 = \tilde{\beta}_t$, λ_t reduces to w_t of Equation 6.22, as it must.

Since every λ_t is positive, Equation 6.24 applies. **If** $\epsilon_\theta(\cdot,t)$ can represent any function of x_t for each t, the idealization of Section 6.9.6, then all these objectives and $\mathcal{L}_{\text{simple}}$ have the same minimizer $\mathbb{E}[\epsilon \mid x_t]$. In this sense the network trained for DDPM can be used for any γ without retraining. A finite network shared across t only approximates this common minimizer. The deterministic case $\gamma_t = 0$ is not covered by this identity, since $\lambda_t \to \infty$ as $\gamma_t \to 0$. J. Song et al. (2021, sec. 4.1) introduce it as the special case $\gamma_t = 0$ of the generative process and note that, although their Theorem 1 does not cover

it, it can be approximated by making γ_t very small; every such γ shares the minimizer above.

6.5.6. Skipping steps

Choose a subsequence $1 = \tau_1 < \cdots < \tau_S = T$ and write Equation 6.34 for each consecutive pair $s = \tau_{i-1} < t = \tau_i$, with $\bar{\alpha}_s$ in place of $\bar{\alpha}_{t-1}$. The proof of Proposition 6.4 uses only the marginals at the two times involved, so every selected time keeps its marginal; J. Song et al. (2021) attach the unselected times directly to x_0 through $q(x_t \mid x_0)$, which keeps the bound valid and plays no role in sampling [App. C.1]. A model trained with $T = 1000$ can therefore sample in $S \ll T$ steps (J. Song et al. 2021, sec. 4.2). For $\eta = 1$ the variance is the posterior variance of the jump: Proposition 6.3 with β_t replaced by the variance $1 - \bar{\alpha}_t/\bar{\alpha}_s$ of the forward transition from s to t, and $\bar{\alpha}_{t-1}$ by $\bar{\alpha}_s$. The marginals remain exact for large jumps, but errors in $\hat{x}_0$ and discretization errors grow with the jump (Section 6.6). Deterministic sampling tolerates large jumps best: with 20 to 100 steps DDIM gives samples comparable to 1000-step models, a 10 to 50 times speed-up (J. Song et al. 2021, sec. 5.1). In full, generalized DDIM sampling on a subsequence is:

1. Choose $1 = \tau_1 < \cdots < \tau_S = T$ and $\eta \in [0, 1]$, and draw $x_{\tau_S} \sim \mathcal{N}(0, I)$.
2. For $i = S, \dots, 1$, with $t = \tau_i$, $s = \tau_{i-1}$ and $\tau_0 := 0$: compute $\hat{\epsilon} = \epsilon_\theta(x_t, t)$ and $\hat{x}_0$ by Equation 6.25, set $\gamma^2 = \eta^2 \frac{1-\bar{\alpha}_s}{1-\bar{\alpha}_t}(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_s})$, and update $x_s = \sqrt{\bar{\alpha}_s}\,\hat{x}_0 + \sqrt{1 - \bar{\alpha}_s - \gamma^2}\,\hat{\epsilon} + \gamma z$ with a fresh $z \sim \mathcal{N}(0, I)$.
3. Return x_0. At the last step $\bar{\alpha}_0 = 1$ gives $\gamma = 0$ and $x_0 = \hat{x}_0$, the counterpart of reporting the decoder mean in DDPM.

The function `ddim` in Section 6.6 implements exactly these steps.

6.5.7. DDPM and DDIM side by side

Table 6.3 collects the differences between the two samplers.

Table 6.3.: DDPM and DDIM compared. The last two rows draw on Section 6.5.4 and Section 6.5.6.

	DDPM	DDIM
Forward structure	Markov chain linking adjacent states only	non-Markovian family q_γ that may depend on x_0
Marginal at each time	Equation 6.6	same as DDPM (Proposition 6.4)
Training	$\epsilon_\theta(x_t, t)$ trained with Equation 6.21	the same $\epsilon_\theta(x_t, t)$ reused
One step	Equation 6.30	Equation 6.36; with $\gamma_t = 0$, Equation 6.38

	DDPM	DDIM
Stochasticity	fresh noise at every step	controlled by $\eta \in [0,1]$; deterministic for $\eta = 0$
Number of steps	all T training steps in Ho et al. (2020); on a subsequence it is DDIM with $\eta = 1$	any subsequence of $S \leq T$ steps

6.6. Reproducible numerical examples

The following computations check the equations numerically and isolate the difference between the DDPM and DDIM samplers using the Bayes-optimal noise predictor. All code uses only the Python standard library, except the Swiss-roll snippet of Section 6.6.3, which uses NumPy. The plotted figures are drawn with TikZ directly from the values these functions return; the drawing code only places those values on axes.

6.6.1. A two-step scalar example

Fix two steps with $x_0 = 2$, $(\beta_1, \beta_2) = (0.1, 0.2)$ and $(\epsilon_1, \epsilon_2) = (0.5, -1)$. Then $(\alpha_1, \alpha_2, \bar{\alpha}_2) = (0.9, 0.8, 0.72)$, and the forward samples are

$$\begin{aligned} x_1 &= \sqrt{0.9}(2) + \sqrt{0.1}(0.5) \approx 2.0555, \\ x_2 &= \sqrt{0.8}(2.0555) + \sqrt{0.2}(-1) \approx 1.3913, \\ \epsilon &= \frac{x_2 - \sqrt{0.72}x_0}{\sqrt{0.28}} \approx -0.5779. \end{aligned}$$

The variance of the two-step noise is $\alpha_2\beta_1 + \beta_2 = 0.28 = 1 - \bar{\alpha}_2$, so the last line is the noise that expresses the same x_2 in one step through Equation 6.7. Substituting the same numbers into Equation 6.16,

$$\begin{aligned} \tilde{\beta}_2 &= \frac{1 - 0.9}{1 - 0.72}(0.2) = \frac{1}{14} \approx 0.07143, \\ \tilde{\mu}_2(x_2, x_0) &= \frac{\sqrt{0.8}(1 - 0.9)(1.3913) + 0.2\sqrt{0.9}(2)}{1 - 0.72} \approx 1.7997. \end{aligned}$$

The actual $x_1 = 2.0555$ on this path lies about 0.96 standard deviations above the posterior mean: the posterior mean is the centre of the conditional distribution, not the latent that happened to be drawn.

Assume an oracle predictor that returns exactly the noise of this path, $\epsilon_\theta(x_2, 2) = \epsilon = -0.5779$. Then Equation 6.25 returns $\hat{x}_0 = 2$ exactly, and three updates can be compared:

- Equation 6.18 (the DDPM mean): $\frac{1}{\sqrt{0.8}}(1.3913 - \frac{0.2}{\sqrt{0.28}}(-0.5779)) \approx 1.7997$, equal to $\tilde{\mu}_2$.
- Equation 6.36 with $\gamma_2^2 = \tilde{\beta}_2$: $\sqrt{0.9}(2) + \sqrt{0.1 - 0.07143}(-0.5779) \approx 1.7997$, equal to the DDPM mean as Proposition 6.5 says.
- Deterministic DDIM ($\gamma_2 = 0$): $\sqrt{0.9}(2) + \sqrt{0.1}(-0.5779) \approx 1.7146$. Instead of adding fresh noise, it moves further along the ϵ direction.

```
import math
sq = math.sqrt
x0, b1, b2, e1, e2 = 2.0, 0.1, 0.2, 0.5, -1.0
a1, a2 = 1 - b1, 1 - b2
ab1, ab2 = a1, a1 * a2
x1 = sq(a1) * x0 + sq(b1) * e1                    # forward step 1
x2 = sq(a2) * x1 + sq(b2) * e2                    # forward step 2
eps = (x2 - sq(ab2) * x0) / sq(1 - ab2)           # one-step noise
bt = (1 - ab1) / (1 - ab2) * b2                   # beta_tilde_2
mu = (sq(a2) * (1 - ab1) * x2 + b2 * sq(ab1) * x0) / (1 - ab2)
mu_eps = (x2 - b2 / sq(1 - ab2) * eps) / sq(a2)   # epsilon form
x0_hat = (x2 - sq(1 - ab2) * eps) / sq(ab2)       # exact eps given
ddim0 = sq(ab1) * x0_hat + sq(1 - ab1) * eps      # gamma = 0
ddimb = sq(ab1) * x0_hat + sq(1 - ab1 - bt) * eps  # gamma^2 = bt
z = (x1 - mu) / sq(bt)
print(f"x1={x1:.4f} x2={x2:.4f} eps={eps:.4f} beta_tilde={bt:.5f}")
print(f"mu={mu:.4f} mu_eps={mu_eps:.4f} z={z:.3f}")
print(f"x0_hat={x0_hat:.4f} ddim0={ddim0:.4f} ddim_bt={ddimb:.4f}")
# x1=2.0555 x2=1.3913 eps=-0.5779 beta_tilde=0.07143
# mu=1.7997 mu_eps=1.7997 z=0.957
# x0_hat=2.0000 ddim0=1.7146 ddim_bt=1.7997
```

6.6.2. DDPM and DDIM on a one-dimensional mixture

Using the Bayes-optimal noise predictor $\epsilon^*(x_t, t) = \mathbb{E}[\epsilon \mid x_t]$ (Equation 6.24), computed analytically, instead of a trained network isolates the properties of the samplers from training error. Note that this is the conditional mean of the noise, not the noise actually added on a path. Take the data to be the one-dimensional Gaussian mixture $x_0 \sim 0.3\,\mathcal{N}(-1.5, 0.2^2) + 0.7\,\mathcal{N}(1, 0.3^2)$. Blurring each component with Equation 6.6 gives $q_t(x) = \sum_k \pi_k \mathcal{N}(x; \sqrt{\bar{\alpha}_t}\mu_k, V_{t,k})$ with $V_{t,k} = \bar{\alpha}_t s_k^2 + 1 - \bar{\alpha}_t$. With the responsibilities $r_k(x) = \pi_k \mathcal{N}(x; \sqrt{\bar{\alpha}_t}\mu_k, V_{t,k})/q_t(x)$, Equation 6.31 gives

$$\epsilon^*(x,t) = -\sqrt{1-\bar{\alpha}_t}\,\nabla \log q_t(x) = \sqrt{1-\bar{\alpha}_t}\sum_k r_k(x)\frac{x-\sqrt{\bar{\alpha}_t}\mu_k}{V_{t,k}}.$$

With the linear schedule ($T = 1000$), $N = 4000$ draws $x_T \sim \mathcal{N}(0, I)$ were run through 1000-step DDPM ($\sigma_t^2 = \tilde{\beta}_t$) and through deterministic DDIM on an evenly spaced subsequence of S steps. Table 6.4 reports the fraction P in the right mode and the mean / standard deviation of each mode (boundary $x = -0.25$).

Table 6.4.: Final samples on the one-dimensional mixture with the Bayes-optimal predictor ($N = 4000$). Each mode entry is mean / standard deviation.

Sampler	Steps S	P(right)	Left mode	Right mode
Target	—	0.700	−1.500 / 0.200	+1.000 / 0.300
DDPM	1000	0.706	−1.487 / 0.200	+1.003 / 0.293
DDIM, $\eta = 0$	50	0.701	−1.479 / 0.191	+0.991 / 0.286
DDIM, $\eta = 0$	10	0.721	−1.404 / 0.230	+0.932 / 0.263

With $N = 4000$ the standard error of P is about 0.007. Three observations follow.

1. **Both samplers closely approximate the same target.** With ϵ^*, 1000-step DDPM and 50-step deterministic DDIM ($\eta = 0$) reproduce the target mode fraction to within one standard error, and the mode means and standard deviations to within 0.021 (Figure 6.11 (c)). The remaining discrepancies come from the Gaussian reverse-transition approximation in finite-step DDPM, the discretization error of finite-step DDIM, the small mismatch between q_T and $\mathcal{N}(0, I)$, and Monte Carlo error. The experiment therefore supports reuse of the same ideal noise predictor without claiming that either finite-step sampler is exact.
2. **DDIM is deterministic.** Started twice from the same $x_T = 0.3$, DDIM returns 0.9608 both times, while DDPM returns different values each time, here −1.4437 and 1.3192, in different modes. In Figure 6.11 (a)–(b), DDPM paths jitter at every step and can reach different modes from the same start, while DDIM paths are smooth and, for these 50 steps, keep the order of their starting points.
3. **Too few steps fail.** At $S = 10$ deterministic DDIM errs clearly: the right-mode fraction is 0.721 against 0.700, the left-mode mean −1.404 against −1.500, and the left-mode spread 0.230 against 0.200. The inference family keeps exact marginals on any subsequence (Section 6.5.6), but the generative chain substitutes $\hat{x}_0$ for x_0, and the resulting error grows with the size of each jump.

Dropping the noise of DDPM. Running the DDPM mean with $z = 0$ at every step, from the first 400 of the same draws x_T, sends every sample to $x_0 \in [0.955, 0.965]$: without the injected noise the chain here contracts to almost a single point near the larger mode, which is why the sampler of Section 6.4.6 adds $\sigma_t z$.

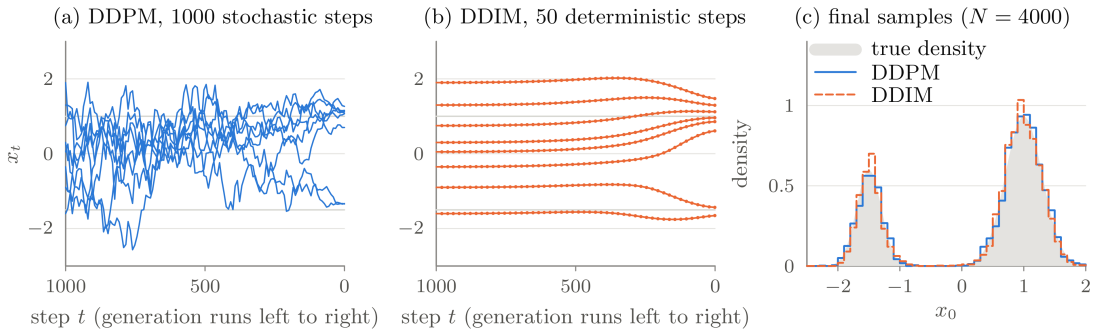


Figure 6.11.: One noise predictor, two samplers, the same target. With the Bayes-optimal predictor ϵ^* on the mixture of Table 6.4: (a) eight paths of 1000-step DDPM jitter because fresh noise enters at every step; (b) 50-step deterministic DDIM paths from the same eight starts are smooth, with dots where ϵ^* is evaluated; (c) histograms (45 bins on $[-2.5, 2]$) of the $N = 4000$ final samples of both samplers match the true density (grey). Grey horizontal lines in (a) and (b) mark the mode means -1.5 and 1, and generation runs from left to right. All plotted values are returned by `figure_data()` in the code below.

The following code reproduces Table 6.4, the determinism check, the $z = 0$ experiment and panel (b) of Figure 6.5 exactly (about 20 seconds with CPython 3.12); its output is listed in the final comments, and `figure_data()` returns the paths and histograms of Figure 6.11.

```
import math, random

T = 1000
# beta[t-1] = beta_t, linear from 1e-4 to 0.02
beta = [1e-4 + (0.02 - 1e-4) * i / (T - 1) for i in range(T)]
abar = [1.0]                                    # abar[t], abar[0] = 1
for b in beta:
    abar.append(abar[-1] * (1 - b))

# data: 0.3 N(-1.5, 0.2^2) + 0.7 N(1, 0.3^2)
PI, MU, SD = (0.3, 0.7), (-1.5, 1.0), (0.2, 0.3)

def eps_star(x, t):
    """Exact E[eps | x_t = x] = -sqrt(1 - abar_t) * score."""
    a, logw, z = abar[t], [], []
    for p, m, s in zip(PI, MU, SD):
        var = a * s * s + 1 - a           # Var(x_t | component)
        d = x - math.sqrt(a) * m
```

```
        lp = math.log(p) - 0.5 * math.log(var) - d * d / (2 * var)
        logw.append(lp)
        z.append(d / var)
    w = [math.exp(l - max(logw)) for l in logw]
    num = sum(wi * zi for wi, zi in zip(w, z))
    return math.sqrt(1 - a) * num / sum(w)

def ddpm(x, rng, path=None):
    """Ancestral sampling with sigma_t^2 = beta_tilde_t."""
    for t in range(T, 0, -1):
        b = beta[t - 1]
        e = eps_star(x, t)
        mean = (x - b / math.sqrt(1 - abar[t]) * e)
        mean /= math.sqrt(1 - b)
        var = (1 - abar[t - 1]) / (1 - abar[t]) * b
        if t > 1:                              # z = 0 at t = 1
            mean += math.sqrt(var) * rng.gauss(0, 1)
        x = mean
        if path is not None: path.append((t - 1, x))
    return x

def ddim(x, steps=50, eta=0.0, rng=None, path=None):
    """Generalized DDIM on an evenly spaced sub-sequence of 1..T."""
    taus = [round(1 + i * (T - 1) / (steps - 1))
            for i in range(steps)]
    for i in range(steps - 1, -1, -1):
        t, s = taus[i], (taus[i - 1] if i > 0 else 0)
        e = eps_star(x, t)
        x0_hat = (x - math.sqrt(1 - abar[t]) * e)
        x0_hat /= math.sqrt(abar[t])
        r = 1 - abar[t] / abar[s]
        g2 = eta ** 2 * (1 - abar[s]) / (1 - abar[t]) * r
        k = math.sqrt(1 - abar[s] - g2)
        x = math.sqrt(abar[s]) * x0_hat + k * e
        if g2 > 0: x += math.sqrt(g2) * rng.gauss(0, 1)
        if path is not None: path.append((s, x))
    return x

def summary(xs):
    def ms(xs):
        m = sum(xs) / len(xs)
        return m, math.sqrt(sum((u - m) ** 2 for u in xs) / len(xs))
    L = [u for u in xs if u <= -0.25]
```

```
    R = [u for u in xs if u > -0.25]
    return len(R) / len(xs), ms(L), ms(R)

def report(name, xs):
    p, (mL, sL), (mR, sR) = summary(xs)
    print(f"{name:13s} P={p:.3f}  L {mL:+.3f}/{sL:.3f}"
          f"  R {mR:+.3f}/{sR:.3f}")

def figure_data():
    """Paths and histograms drawn in the figure."""
    starts = [-1.6, -0.9, -0.35, 0.05, 0.3, 0.75, 1.3, 1.9]
    prng, paths_ddpm, paths_ddim = random.Random(7), [], []
    for x in starts:
        p = [(T, x)]
        ddpm(x, prng, p)
        paths_ddpm.append(p[::8] + [p[-1]])  # every 8th step
        q = [(T, x)]
        ddim(x, 50, path=q)
        paths_ddim.append(q)
    rng = random.Random(0)
    xT = [rng.gauss(0, 1) for _ in range(4000)]
    s_ddpm = [ddpm(x, rng) for x in xT]   # same draws as the table
    s_ddim = [ddim(x, 50) for x in xT]
    lo, hi, nb = -2.5, 2.0, 45
    def hist(xs):                          # density per bin
        cnt = [0] * nb
        for x in xs:
            i = int((x - lo) / (hi - lo) * nb)
            if 0 <= i < nb: cnt[i] += 1
        return [n / (len(xs) * (hi - lo) / nb) for n in cnt]
    return paths_ddpm, paths_ddim, hist(s_ddpm), hist(s_ddim)

def vlb_terms(n=1401):
    """Interior VLB terms L_{t-1} (nats), t = 2..T, with eps*."""
    h, out = 14 / (n - 1), []
    for t in range(2, T + 1):
        a, e2 = abar[t], 0.0
        for i in range(n):
            x = -7 + i * h
            q = sum(p * math.exp(-(x - math.sqrt(a) * m) ** 2
                                 / (2 * (a * s * s + 1 - a)))
                    / math.sqrt(2 * math.pi * (a * s * s + 1 - a))
                    for p, m, s in zip(PI, MU, SD))
```

```
            e2 += q * eps_star(x, t) ** 2 * h
        b = beta[t - 1]
        out.append(b / (2 * (1 - b) * (1 - abar[t - 1])) * (1 - e2))
    return out

def ddpm_mean_only(x):
    """DDPM with the noise term dropped: z = 0 at every step."""
    for t in range(T, 0, -1):
        b = beta[t - 1]
        e = eps_star(x, t)
        x = (x - b / math.sqrt(1 - abar[t]) * e) / math.sqrt(1 - b)
    return x

if __name__ == "__main__":
    rng, N = random.Random(0), 4000
    xT = [rng.gauss(0, 1) for _ in range(N)]
    print("target        P=0.700  L -1.500/0.200  R +1.000/0.300")
    report("DDPM, T=1000", [ddpm(x, rng) for x in xT])
    for steps in (50, 10):
        report(f"DDIM, S={steps}", [ddim(x, steps) for x in xT])
    d1, d2 = ddim(0.3), ddim(0.3)
    p1, p2 = ddpm(0.3, rng), ddpm(0.3, rng)
    print("x_T = 0.3: DDIM", round(d1, 4), round(d2, 4),
          "| DDPM", round(p1, 4), round(p2, 4))
    zs = [ddpm_mean_only(x) for x in xT[:400]]
    print("z = 0 throughout: x_0 in [%.3f, %.3f]"
          % (min(zs), max(zs)))
    L = vlb_terms()
    print("VLB terms: L_1 %.2f, L_999 %.1e" % (L[0], L[998]),
          "| share of t<=10: %.1f%%" % (100 * sum(L[:9]) / sum(L)))
# target        P=0.700  L -1.500/0.200  R +1.000/0.300
# DDPM, T=1000  P=0.706  L -1.487/0.200  R +1.003/0.293
# DDIM, S=50    P=0.701  L -1.479/0.191  R +0.991/0.286
# DDIM, S=10    P=0.721  L -1.404/0.230  R +0.932/0.263
# x_T = 0.3: DDIM 0.9608 0.9608 | DDPM -1.4437 1.3192
# z = 0 throughout: x_0 in [0.955, 0.965]
# VLB terms: L_1 0.60, L_999 5.7e-07 | share of t<=10: 42.6%
```

6.6.3. The Swiss roll

Figure 6.9 uses the exact score of an empirical distribution: with data points $d_1, \dots, d_n$, the noisy marginal q_t is a mixture of Gaussians centred at $\sqrt{\bar{\alpha}_t} d_i$, and ϵ^* follows from

Equation 6.31 as a softmax-weighted average. The following NumPy code returns the plotted points and arrows (about 40 seconds). Its check, listed in the final comment, confirms the caption: the median distance from a generated sample at $t = 0$ to the nearest training point is 0.0045, far below the 0.03 jitter of the data.

```
import math
import numpy as np

T = 1000
beta = np.linspace(1e-4, 0.02, T)
abar = np.concatenate([[1.0], np.cumprod(1 - beta)])

def roll(n, rng):
    th = 1.5 * np.pi + 3 * np.pi * rng.random(n)
    p = np.stack([th * np.cos(th), th * np.sin(th)], 1) / 6
    return p + 0.03 * rng.standard_normal(p.shape)

data = roll(1500, np.random.default_rng(0))   # empirical law

def eps_star(x, t):
    """E[eps | x_t = x] under the empirical data law."""
    a, b2 = math.sqrt(abar[t]), 1 - abar[t]
    d2 = ((x[:, None] - a * data[None]) ** 2).sum(-1)
    w = np.exp(-(d2 - d2.min(1, keepdims=True)) / (2 * b2))
    w /= w.sum(1, keepdims=True)
    return (x - a * w @ data) / math.sqrt(b2)

def mean(x, t):                                  # DDPM reverse mean
    e = eps_star(x, t)
    return (x - beta[t - 1] / math.sqrt(1 - abar[t]) * e) \
        / math.sqrt(1 - beta[t - 1])

def swiss_roll_figure(times=(0, 60, T)):
    """Forward samples, reverse samples and drift field."""
    x0 = roll(600, np.random.default_rng(1))
    z = np.random.default_rng(2).standard_normal(x0.shape)
    fwd = {t: math.sqrt(abar[t]) * x0 + math.sqrt(1 - abar[t]) * z
           for t in times}
    g = np.random.default_rng(3)
    x = g.standard_normal((600, 2))
    rev = {T: x.copy()}
    for t in range(T, 0, -1):
        var = (1 - abar[t - 1]) / (1 - abar[t]) * beta[t - 1]
        x = mean(x, t)
```

```
        if t > 1:
            x = x + math.sqrt(var) * g.standard_normal(x.shape)
        if t - 1 in times:
            rev[t - 1] = x.copy()
    s = np.linspace(-2.4, 2.4, 13)
    grid = np.stack(np.meshgrid(s, s), -1).reshape(-1, 2)
    drift = {t: mean(grid, max(t, 1)) - grid for t in times}
    return fwd, rev, drift

if __name__ == "__main__":
    fwd, rev, drift = swiss_roll_figure()
    d = np.sqrt(((rev[0][:, None] - data[None]) ** 2).sum(-1))
    print("t=0 samples: median distance to nearest data point",
          round(float(np.median(d.min(1))), 4))
# t=0 samples: median distance to nearest data point 0.0045
```

6.6.4. Schedule and weight statistics

The schedule and weight values quoted in Section 6.4.1.3 (including the cosine schedule), Section 6.4.2.2, Section 6.4.2.5 and Table 6.2 are reproduced by the following code, whose output is listed in the final comments; `curves()` returns the curves plotted in Figure 6.2, Figure 6.6 (b) and Figure 6.5 (a).

```
import math

T = 1000
beta = [1e-4 + (0.02 - 1e-4) * i / (T - 1) for i in range(T)]
abar = [1.0]
for b in beta:
    abar.append(abar[-1] * (1 - b))

def snr(t):
    return abar[t] / (1 - abar[t])

def ratio(t):           # beta_t / (alpha_t (1 - abar_{t-1}))
    return beta[t - 1] / ((1 - beta[t - 1]) * (1 - abar[t - 1]))

def curves(TT=1000):
    """beta_t, abar_t, SNR(t) and tilde-beta_t / beta_t of the
    linear schedule shortened to TT steps (ends times 1000/TT)."""
    sc = 1000 / TT
    b = [sc * (1e-4 + (0.02 - 1e-4) * i / (TT - 1))
```

```
          for i in range(TT)]
    ab = [1.0]
    for x in b:
        ab.append(ab[-1] * (1 - x))
    snr_t = [ab[t] / (1 - ab[t]) for t in range(1, TT + 1)]
    ratio_t = [(1 - ab[t - 1]) / (1 - ab[t])
               for t in range(1, TT + 1)]
    return b, ab, snr_t, ratio_t

def cosine_abar(t, s=0.008):   # Nichol & Dhariwal, Sec. 3.2
    f = lambda u: math.cos((u / T + s) / (1 + s) * math.pi / 2) ** 2
    return f(t) / f(0)

half = min(range(1, T + 1), key=lambda t: abs(abar[t] - 0.5))
print("abar_t closest to 1/2 at t =", half,
      "| abar_T =", f"{abar[T]:.1e}")
print("cosine: abar_t closest to 1/2 at t =",
      min(range(1, T + 1), key=lambda t: abs(cosine_abar(t) - 0.5)))
print("SNR(1) =", f"{snr(1):.0f}", "| SNR(T) =", f"{snr(T):.1e}")
print("L_T/d bound:", f"{-0.5 * math.log(1 - abar[T]):.1e} nats")
print("ratio at t = 2:", round(ratio(2), 2))
ts = range(2, T + 1)    # interior terms t = 2..T
w = {"tilde": [ratio(t) / 2 for t in ts],
     "beta": [beta[t - 1] / (2 * (1 - beta[t - 1]) * (1 - abar[t]))
              for t in ts],
     "simple": [1.0 for t in ts]}
for k, vals in w.items():
    m, tot = sum(vals) / len(vals), sum(vals)
    rel = [round(vals[t - 2] / m, 2) for t in (2, 10, 100, 500)]
    print(f"{k:6s}", rel, f"t<=10 {sum(vals[:9]) / tot:.1%}",
          f"t<=100 {sum(vals[:99]) / tot:.1%}")
x = [math.log(t) for t in range(2, 101)]
y = [math.log(w["tilde"][t - 2]) for t in range(2, 101)]
mx, my = sum(x) / len(x), sum(y) / len(y)
num = sum((a - mx) * (b - my) for a, b in zip(x, y))
print("log-log slope, t = 2..100:",
      round(num / sum((a - mx) ** 2 for a in x), 2))
# abar_t closest to 1/2 at t = 259 | abar_T = 4.0e-05
# cosine: abar_t closest to 1/2 at t = 496
# SNR(1) = 9999 | SNR(T) = 4.0e-05
# L_T/d bound: 2.0e-05 nats
# ratio at t = 2: 1.2
# tilde  [58.9, 8.49, 1.01, 0.54] t<=10 18.4% t<=100 39.2%
```

```
# beta   [29.1, 7.87, 1.08, 0.59] t<=10 12.8% t<=100 34.1%
# simple [1.0, 1.0, 1.0, 1.0] t<=10 0.9% t<=100 9.9%
# log-log slope, t = 2..100: -0.96
```

6.7. Summary

1. Markov forward transitions and Gaussian closure give $q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$, the one-step renoising formula (Proposition 6.2).
2. The reverse process is approximated by Gaussian Markov transitions for small steps, and the upper bound on the negative log-likelihood decomposes, by Bayes' rule and a telescoping sum, into $L_T + \sum_{t\geq 2} L_{t-1} + L_0$.
3. Completing the square gives the exact posterior $\mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t I)$ (Proposition 6.3); rewriting $\tilde{\mu}_t$ with the one-step renoising formula turns mean prediction into the noise-estimation loss. Training is self-supervised.
4. In the fixed-variance DDPM considered here, ancestral sampling adds fresh noise $\sigma_t z$ to the mean, with σ_t determined in closed form by the forward schedule; learned-variance variants relax this choice.
5. DDIM uses a non-Markovian family with the same marginals as DDPM, and keeps the same loss and the same noise estimator.
6. The DDPM reverse chain is always Markov. A Gaussian reverse step is, heuristically, accurate when the relative SNR drop per step, $\mathrm{SNR}(t-1)/\mathrm{SNR}(t) - 1$, is small; for the linear schedule the drop is largest in the first steps (about 1.2 at $t = 2$).
7. The simplified loss is a reweighting that sets the VLB coefficients w_t to 1; in x_0-space the VLB weights each step by the SNR it loses, $\frac{1}{2}(\mathrm{SNR}(t-1) - \mathrm{SNR}(t))$ (Equation 6.29). The reweighting improves FID from 13.51 to 3.17 at the cost of NLL rising slightly from 3.70 to 3.75 bits/dim.
8. ϵ-, x_0- and v-prediction are rotated coordinates of the same latent pair, and in x_0-space they weight the error by SNR, 1 and SNR+1.
9. DDIM becomes DDPM for $\gamma_t^2 = \tilde{\beta}_t$ and a deterministic sampler for $\gamma_t = 0$. For every $\gamma > 0$ its bound is a positively weighted noise-prediction MSE, so an ideal noise predictor serves every γ; DDIM can also skip steps along a subsequence, where deterministic sampling degrades least.

6.8. Connections and further reading

- **Latent-variable models.** The variational bound has the same structure as the VAE ELBO (Kingma and Welling 2014, arXiv:1312.6114). DDPM is a deep latent-variable model whose inference path is fixed rather than learned.

- **Continuous time.** Score-based SDEs place stochastic DDPM sampling and deterministic probability-flow sampling in a common continuous-time framework (Y. Song et al. 2021, arXiv:2011.13456).
- **Original papers.** Diffusion probabilistic models start with Sohl-Dickstein et al. (2015, arXiv:1503.03585); the DDPM derivation and the simplified loss are in Ho et al. (2020, arXiv:2006.11239), Secs. 3–4; learned variances, the cosine schedule and the VLB analysis in Nichol and Dhariwal (2021, arXiv:2102.09672); the SNR form of the bound in Kingma et al. (2021, arXiv:2107.00630); DDIM in J. Song et al. (2021, arXiv:2010.02502), Secs. 3–5; v-prediction and loss weighting in Salimans and Ho (2022, arXiv:2202.00512), Sec. 4 and App. D; Noise2Score in Kim and Ye (2021, arXiv:2106.07009); score matching and its denoising form in Hyvarinen (2005, JMLR 6, 695–709) and Vincent (2011, Neural Computation 23, 1661–1674).

6.9. Appendix: proofs

This appendix collects the longer proofs and algebraic checks of results stated in the main text.

6.9.1. Gaussian lemmas

Proof of Lemma 6.1. By independence the characteristic function factorizes, $\mathbb{E}\, e^{ik^{\mathsf{T}}(Au+Bv+c)} = e^{ik^{\mathsf{T}}c}\,\varphi_u(A^{\mathsf{T}}k)\,\varphi_v(B^{\mathsf{T}}k)$, and since $\varphi_u(k) = \exp(ik^{\mathsf{T}}m_u - \tfrac{1}{2}k^{\mathsf{T}}\Sigma_u k)$ the product is $\exp(ik^{\mathsf{T}}(Am_u + Bm_v + c) - \tfrac{1}{2}k^{\mathsf{T}}(A\Sigma_u A^{\mathsf{T}} + B\Sigma_v B^{\mathsf{T}})k)$, the characteristic function of the claimed Gaussian. □

Proof of Lemma 6.2. Expanding the right-hand side gives $-\tfrac{1}{2}x^{\mathsf{T}}Ax + x^{\mathsf{T}}h - \tfrac{1}{2}h^{\mathsf{T}}A^{-1}h + \tfrac{1}{2}h^{\mathsf{T}}A^{-1}h$, which is the left-hand side. The last term does not depend on x and is absorbed into the normalizing constant. □

Proof of Lemma 6.3. $\log q(x) - \log p(x) = \frac{d}{2}\log\frac{s_2^2}{s_1^2} - \frac{\|x-m_1\|^2}{2s_1^2} + \frac{\|x-m_2\|^2}{2s_2^2}$. Under $x \sim q$, $\mathbb{E}\|x-m_1\|^2 = ds_1^2$, and since the cross term has zero mean, $\mathbb{E}\|x-m_2\|^2 = \mathbb{E}\|(x-m_1)+(m_1-m_2)\|^2 = ds_1^2 + \|m_1-m_2\|^2$. Substituting gives Equation 6.3. □

6.9.2. The reverse chain is Markov

Proof of Proposition 6.1. If the forward chain is Markov, then $q(x_{t+1:T} \mid x_t, x_{t-1}) = q(x_{t+1:T} \mid x_t)$, so

$$q(x_{t-1} \mid x_{t:T}) = \frac{q(x_{t-1})\,q(x_t \mid x_{t-1})\,q(x_{t+1:T} \mid x_t)}{q(x_t)\,q(x_{t+1:T} \mid x_t)} = \frac{q(x_{t-1})\,q(x_t \mid x_{t-1})}{q(x_t)} = q(x_{t-1} \mid x_t).$$

The numerator is the chain rule for the joint, and the denominator is $q(x_{t:T}) = q(x_t)\, q(x_{t+1:T} \mid x_t)$. No assumption on the step size is used. □

6.9.3. The Gaussian approximation for small steps (heuristic)

Derivation of Equation 6.9. This is a first-order approximation, not a rigorous proof. By Bayes' rule,

$$q(x_{t-1} \mid x_t) = \frac{q(x_t \mid x_{t-1})\, q_{t-1}(x_{t-1})}{q_t(x_t)} \propto_{x_{t-1}} \exp\left(-\frac{\alpha_t}{2\beta_t}\, \|x_{t-1} - m_t\|^2\right) q_{t-1}(x_{t-1}), \qquad m_t := \frac{x_t}{\sqrt{\alpha_t}},$$

using $\|x_t - \sqrt{\alpha_t} x_{t-1}\|^2 = \alpha_t \|x_{t-1} - m_t\|^2$. The first factor is a Gaussian of width $\sqrt{\beta_t/\alpha_t}$, so for small β_t the mass of x_{t-1} is concentrated near m_t. Expanding there, $\log q_{t-1}(x_{t-1}) = \log q_{t-1}(m_t) + g_t^{\mathsf{T}}(x_{t-1} - m_t) + r_t$ with $g_t := \nabla \log q_{t-1}(m_t)$, where the remainder r_t is of order $\frac{1}{2}\|\nabla^2 \log q_{t-1}\|\, \|x_{t-1} - m_t\|^2$. Dropping r_t and applying Lemma 6.2 with precision $A = (\alpha_t/\beta_t) I$ and linear term $h = (\alpha_t/\beta_t) m_t + g_t$ gives the mean $A^{-1}h = m_t + (\beta_t/\alpha_t) g_t$ and the covariance $(\beta_t/\alpha_t) I$.

Since $\|x_{t-1} - m_t\|^2 \approx d\beta_t/\alpha_t$ in that region, the dropped r_t, compared with the Gaussian exponent $\frac{\alpha_t}{2\beta_t}\|x_{t-1} - m_t\|^2 \approx d/2$, has relative size about $\beta_t \|\nabla^2 \log q_{t-1}\|/\alpha_t$. If the data are sharply concentrated, q_{t-1} is the data blurred by a Gaussian of variance $1 - \bar\alpha_{t-1}$, so $\|\nabla^2 \log q_{t-1}\| \approx 1/(1 - \bar\alpha_{t-1})$. For data with several well-separated modes the curvature between the modes can be larger, so this is a scale rather than a bound. For Gaussian data $x_0 \sim \mathcal{N}(0, s^2 I)$ this can be checked exactly. Then $q_{t-1} = \mathcal{N}(0, \kappa I)$ with $\kappa := \bar\alpha_{t-1} s^2 + 1 - \bar\alpha_{t-1}$, the precision of the exact posterior is $\alpha_t/\beta_t + 1/\kappa$, and with $\delta := \beta_t/(\alpha_t \kappa)$ the exact mean and variance are $m_t/(1+\delta)$ and $(\beta_t/\alpha_t)/(1+\delta)$. Equation 6.9, with $\nabla \log q_{t-1}(m_t) = -m_t/\kappa$, gives the mean $m_t(1-\delta)$ and the variance β_t/α_t. The means differ by $m_t \delta^2/(1+\delta)$, a relative error of δ^2, while the approximate variance is too large by the relative amount δ; both vanish as $\delta \to 0$. As $s \to 0$, $\delta = \beta_t/(\alpha_t(1 - \bar\alpha_{t-1}))$. Since $\mathrm{SNR}(t-1)/\mathrm{SNR}(t) = (1-\bar\alpha_t)/(\alpha_t(1-\bar\alpha_{t-1}))$ and $(1-\bar\alpha_t) - \alpha_t(1-\bar\alpha_{t-1}) = 1 - \alpha_t = \beta_t$, this limit equals $\mathrm{SNR}(t-1)/\mathrm{SNR}(t) - 1$, which is Equation 6.10.

6.9.4. Decomposition of the variational bound

Proof of Theorem 6.1. Start from the joint factorizations and justify each change of measure in turn.

Step 1. From Equation 6.4 and Equation 6.8, $\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} = p(x_T) \prod_{t=1}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}$; taking logarithms gives Equation 6.13.

Step 2. Separate the $t = 1$ term $\log \frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}$ from the sum.

Step 3. For $t \geq 2$ the Markov property gives $q(x_t \mid x_{t-1}) = q(x_t \mid x_{t-1}, x_0)$, and Bayes' rule for the two variables (x_{t-1}, x_t) conditioned on x_0 gives Equation 6.14. Each term with $t \geq 2$ becomes

$$\log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})} = \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} + \log \frac{q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}.$$

At $t = 1$, $x_{t-1} = x_0$ is itself the conditioning variable, so $q(x_0 \mid x_1, x_0)$ degenerates to a point mass and this rewriting is not used. This is why the $t = 1$ term was separated in step 2.

Step 4. Summing the second log-ratio over $t = 2, \dots, T$ telescopes to $\log q(x_1 \mid x_0) - \log q(x_T \mid x_0)$. Combined with the $t = 1$ term of step 2, $\log q(x_1 \mid x_0)$ cancels, which gives the displayed equation in the main text.

Step 5. Substituting into Equation 6.13 and grouping $-\log p(x_T) + \log q(x_T \mid x_0) = -\log \frac{p(x_T)}{q(x_T \mid x_0)}$,

$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_q \left[-\log \frac{p(x_T)}{q(x_T \mid x_0)} - \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} - \log p_\theta(x_0 \mid x_1) \right].$$

Step 6. The integrand of each term with $t \geq 2$ depends only on (x_0, x_{t-1}, x_t). Marginalizing $q(x_{1:T} \mid x_0)$ onto (x_{t-1}, x_t) gives $q(x_{t-1}, x_t \mid x_0) = q(x_t \mid x_0)\, q(x_{t-1} \mid x_t, x_0)$, so

$$\begin{aligned} \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[\log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} \right] &= \mathbb{E}_{q(x_t \mid x_0)} \left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)}\, dx_{t-1} \right] \\ &= \mathbb{E}_{q(x_t \mid x_0)} [D_{\text{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))]. \end{aligned}$$

In the same way the first term becomes $D_{\text{KL}}(q(x_T \mid x_0) \,\|\, p(x_T))$. Averaging over $x_0 \sim q_{\text{data}}$ gives Equation 6.12. □

6.9.5. The constant term when the variances differ

Proof of Equation 6.17. Substituting $m_1 = \tilde{\mu}_t$, $s_1^2 = \tilde{\beta}_t$, $m_2 = \mu_\theta$, $s_2^2 = \sigma_t^2$ into Lemma 6.3,

$$D_{\text{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)) = \frac{\|\tilde{\mu}_t - \mu_\theta\|^2}{2\sigma_t^2} + \underbrace{\frac{d}{2} \left[\frac{\tilde{\beta}_t}{\sigma_t^2} - 1 + \log \frac{\sigma_t^2}{\tilde{\beta}_t} \right]}_{C_t}.$$

With $\xi := \tilde{\beta}_t/\sigma_t^2$ the bracket is $\xi - 1 - \log \xi \geq 0$, with equality only at $\xi = 1$, so $C_t \geq 0$, with equality only when $\sigma_t^2 = \tilde{\beta}_t$. Taking the expectation over q gives Equation 6.17. □

6.9.6. An optimum that does not depend on the weights

Proof of Equation 6.24. For fixed t and any function f, the tower property makes the cross term $\mathbb{E}[(\epsilon - \mathbb{E}[\epsilon \mid x_t])^{\mathsf{T}}(\mathbb{E}[\epsilon \mid x_t] - f(x_t))]$ vanish, so

$$\mathbb{E}\|\epsilon - f(x_t)\|^2 = \mathbb{E}\|\epsilon - \mathbb{E}[\epsilon \mid x_t]\|^2 + \mathbb{E}\|\mathbb{E}[\epsilon \mid x_t] - f(x_t)\|^2.$$

The second term vanishes only for $f = \mathbb{E}[\epsilon \mid x_t]$. If $\epsilon_\theta(\cdot, t)$ can represent an arbitrary function independently for each t, the minimizer of the sum weighted by $\lambda_t > 0$ is the same $\mathbb{E}[\epsilon \mid x_t]$ at every time. □

6.9.7. The score identity and Tweedie's formula

Proof of Equation 6.31. Exchanging differentiation and integration (justified by dominated convergence, since for the Gaussian kernel $\nabla_{x_t} q(x_t \mid x_0)$ is bounded uniformly in x_0; no moment condition on the data is needed),

$$\nabla \log q_t(x_t) = \frac{\int q_{\text{data}}(x_0)\, q(x_t \mid x_0)\, \nabla_{x_t} \log q(x_t \mid x_0)\, dx_0}{q_t(x_t)} = \mathbb{E}\left[\nabla_{x_t} \log q(x_t \mid x_0) \,\middle|\, x_t\right].$$

From Equation 6.6, $\nabla_{x_t} \log q(x_t \mid x_0) = -(x_t - \sqrt{\bar{\alpha}_t} x_0)/(1 - \bar{\alpha}_t) = -\epsilon/\sqrt{1 - \bar{\alpha}_t}$, which gives the first identity, and the optimal predictor is $\epsilon^* = \mathbb{E}[\epsilon \mid x_t]$ of Equation 6.24. The second identity follows by taking the conditional expectation given x_t of $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$, which gives $\mathbb{E}[x_0 \mid x_t] = (x_t - \sqrt{1 - \bar{\alpha}_t}\, \mathbb{E}[\epsilon \mid x_t])/\sqrt{\bar{\alpha}_t}$, and substituting the first identity. Equation 6.32 follows directly by substituting $\epsilon_\theta = -\sqrt{1 - \bar{\alpha}_t}\, s_\theta$ into Equation 6.19, since $-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta = \beta_t s_\theta$. The one-step identity Equation 6.33 follows from the same argument: exchanging differentiation and integration in $q_t(x_t) = \int q_{t-1}(x_{t-1})\, \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, \beta_t I)\, dx_{t-1}$ gives

$$\nabla \log q_t(x_t) = \mathbb{E}\left[-\frac{x_t - \sqrt{\alpha_t} x_{t-1}}{\beta_t} \,\middle|\, x_t\right] = -\frac{x_t - \sqrt{\alpha_t}\, \mathbb{E}[x_{t-1} \mid x_t]}{\beta_t},$$

which is solved for $\mathbb{E}[x_{t-1} \mid x_t]$. □

6.9.8. The DDIM objective is a weighted noise-prediction loss

Proof of Equation 6.40. This proves Theorem 1 of J. Song et al. (2021, App. B) in this note's notation. Their generative model has $p(x_T) = \mathcal{N}(0, I)$, the transitions $p_\theta(x_{t-1} \mid x_t) := q_\gamma(x_{t-1} \mid x_t, \hat{x}_0(x_t, t))$ for $t \geq 2$, which is the step Equation 6.36, and a Gaussian decoder $p_\theta(x_0 \mid x_1) = \mathcal{N}(\hat{x}_0(x_1, 1), \gamma_1^2 I)$ with $\gamma_1 > 0$ (Sec. 3.2, Eq. 10). Since $q_\gamma(x_{1:T} \mid x_0) = q_\gamma(x_T \mid x_0) \prod_{t=2}^{T} q_\gamma(x_{t-1} \mid x_t, x_0)$ is already written in reverse order,

$$\log \frac{q_\gamma(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})} = \log \frac{q_\gamma(x_T \mid x_0)}{p(x_T)} + \sum_{t=2}^{T} \log \frac{q_\gamma(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} - \log p_\theta(x_0 \mid x_1),$$

and taking expectations as in step 6 of Section 6.9.4 turns the first two groups into KL divergences. The first does not depend on θ.

For $t \geq 2$ write the mean of Equation 6.34 as $c_t x_0 + e_t x_t$ with $e_t := \sqrt{1 - \bar{\alpha}_{t-1} - \gamma_t^2} / \sqrt{1 - \bar{\alpha}_t}$. The generative mean is $c_t \hat{x}_0 + e_t x_t$, so the two means differ by $c_t(x_0 - \hat{x}_0)$. From $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$ and Equation 6.25, $x_0 - \hat{x}_0 = (\epsilon_\theta(x_t, t) - \epsilon) / \sqrt{\mathrm{SNR}(t)}$. Both covariances are $\gamma_t^2 I$, so Lemma 6.3 gives the KL divergence $c_t^2 \|x_0 - \hat{x}_0\|^2 / (2\gamma_t^2) = \lambda_t \|\epsilon - \epsilon_\theta\|^2$ with λ_t of Equation 6.40. By Proposition 6.4, $x_t \sim q_\gamma(x_t \mid x_0)$ is the one-step renoising of x_0 for every γ, so the expectation runs over the same (x_0, ϵ) as in $\mathcal{L}_{\text{simple}}$. In the same way $-\log p_\theta(x_0 \mid x_1) = \|\epsilon - \epsilon_\theta(x_1, 1)\|^2 / (2\gamma_1^2 \,\mathrm{SNR}(1)) + \text{const}$. Altogether the bound equals $\sum_{t=1}^{T} \lambda_t \, \mathbb{E}\|\epsilon - \epsilon_\theta(x_t, t)\|^2 + C$ with $\lambda_1 := 1 / (2\gamma_1^2 \,\mathrm{SNR}(1))$ and a constant C.

Positivity. c_t increases with γ_t. At $\gamma_t = 0$, multiplying by $\sqrt{1 - \bar{\alpha}_t}$ shows that $c_t > 0$ is equivalent to $\bar{\alpha}_{t-1}(1 - \bar{\alpha}_t) > (1 - \bar{\alpha}_{t-1})\bar{\alpha}_t$, that is, to $\bar{\alpha}_{t-1} > \bar{\alpha}_t$, which holds because $\alpha_t < 1$. Hence $\lambda_t > 0$ for every $\gamma_t > 0$.

Check. For $\gamma_t^2 = \tilde{\beta}_t$, Section 6.9.9 gives $c_t = \sqrt{\bar{\alpha}_{t-1}} \beta_t / (1 - \bar{\alpha}_t)$. With $\tilde{\beta}_t = (1 - \bar{\alpha}_{t-1})\beta_t / (1 - \bar{\alpha}_t)$ and $\bar{\alpha}_{t-1} / \bar{\alpha}_t = 1 / \alpha_t$,

$$\lambda_t = \frac{\bar{\alpha}_{t-1} \beta_t^2}{(1 - \bar{\alpha}_t)^2} \cdot \frac{1 - \bar{\alpha}_t}{2(1 - \bar{\alpha}_{t-1})\beta_t} \cdot \frac{1 - \bar{\alpha}_t}{\bar{\alpha}_t} = \frac{\beta_t}{2\alpha_t (1 - \bar{\alpha}_{t-1})} = w_t. \qquad \square$$

6.9.9. Reduction of DDIM to DDPM

Proof of Proposition 6.5. The variances agree by definition, so it suffices to compare the coefficients of the mean. Using $\tilde{\beta}_t = (1 - \bar{\alpha}_{t-1})\beta_t / (1 - \bar{\alpha}_t)$ and $1 - \bar{\alpha}_t - \beta_t = \alpha_t - \bar{\alpha}_t = \alpha_t (1 - \bar{\alpha}_{t-1})$,

$$1 - \bar{\alpha}_{t-1} - \tilde{\beta}_t = (1 - \bar{\alpha}_{t-1}) \frac{1 - \bar{\alpha}_t - \beta_t}{1 - \bar{\alpha}_t} = \frac{\alpha_t (1 - \bar{\alpha}_{t-1})^2}{1 - \bar{\alpha}_t},$$

$$\sqrt{1 - \bar{\alpha}_{t-1} - \tilde{\beta}_t} = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{\sqrt{1 - \bar{\alpha}_t}}.$$

So the coefficient of x_t in the mean of Equation 6.34 is $\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)$, as in Equation 6.16, and, using $\sqrt{\alpha_t}\sqrt{\bar{\alpha}_t} = \alpha_t\sqrt{\bar{\alpha}_{t-1}}$, the coefficient of x_0 is

$$\sqrt{\bar{\alpha}_{t-1}} - \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t} = \sqrt{\bar{\alpha}_{t-1}}\,\frac{(1-\bar{\alpha}_t)-\alpha_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t},$$

which also agrees. Both generative steps substitute the same $\hat{x}_0$ for x_0, and $(x_t - \sqrt{\bar{\alpha}_t}\hat{x}_0)/\sqrt{1-\bar{\alpha}_t} = \epsilon_\theta$, so Equation 6.37 has the same mean as Equation 6.30. □

Based on the scribe note of Hyewon Park, with material from Donghyeon Kim, Seojeong Gu and Chankyu Lee — the four scribes of lecture 06; edited by Jong Chul Ye.

7. Score-based Generative Models with SDEs

> **Not yet written**
>
> Lecture 07 of AI618, given 2026-10-06. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

8. Conditional Diffusion Models

Not yet written

Lecture 08 of AI618, given 2026-10-08. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

9. Latent Diffusion

Not yet written

Lecture 09 of AI618, given 2026-10-13. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

10. Schrodinger Bridge Problems

> **Not yet written**
>
> Lecture 10 of AI618, given 2026-10-15. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

11. Acceleration: High-order Solvers and Consistency Models

> **Not yet written**
>
> Lecture 11 of AI618, given 2026-10-27. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

12. Image Editing and Inverse Problems

> **Not yet written**
>
> Lecture 12 of AI618, given 2026-10-29. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

13. Video and 3D Diffusion

> **Not yet written**
>
> Lecture 13 of AI618, given 2026-11-03. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

Part III.

Flow Models

14. Flow Matching

> **Not yet written**
>
> Lecture 14 of AI618, given 2026-11-05. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

15. Rectified Flow

> **Not yet written**
>
> Lecture 15 of AI618, given 2026-11-10. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

16. Relation between Flow and Diffusion Models

Not yet written

Lecture 16 of AI618, given 2026-11-12. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

17. Probability Paths and the Kolmogorov Equation

> **Not yet written**
>
> Lecture 17 of AI618, given 2026-11-17. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

Part IV.

Discrete Diffusion

18. Discrete Flow Matching

> **Not yet written**
>
> Lecture 18 of AI618, given 2026-11-19. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

19. Diffusion Language Models

Not yet written

Lecture 19 of AI618, given 2026-11-24. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

20. Diffusion Duality

> **Not yet written**
>
> Lecture 20 of AI618, given 2026-11-26. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

21. LLaDA and MMaDA

Not yet written

Lecture 21 of AI618, given 2026-12-01. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

Part V.

Alignment with Reinforcement Learning

22. RL Review: RLHF, DPO, DDPO and GRPO

Not yet written

Lecture 22 of AI618, given 2026-12-03. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

23. Diffusion DPO, DDPO, DanceGRPO and FlowGRPO

Not yet written

Lecture 23 of AI618, given 2026-12-08. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

24. DiffusionNFT, Reward Score Matching and Reward Adjoint Matching

> **Not yet written**
>
> Lecture 24 of AI618, given 2026-12-10. This chapter is merged from the scribe notes for that lecture; until those are written and graded it stands empty.

References

Alain, Guillaume, and Yoshua Bengio. 2014. "What Regularized Auto-Encoders Learn from the Data-Generating Distribution." *Journal of Machine Learning Research* 15: 3563–93.

Alharbi, Yazeed, and Peter Wonka. 2020. "Disentangled Image Generation Through Structured Noise Injection." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/2004.12411.

Arjovsky, Martin, and Léon Bottou. 2017. "Towards Principled Methods for Training Generative Adversarial Networks." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1701.04862.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. 2017. "Wasserstein Generative Adversarial Networks." *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1701.07875.

Arora, Sanjeev, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. 2017. "Generalization and Equilibrium in Generative Adversarial Nets (GANs)." *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1703.00573.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. *Layer Normalization*. https://arxiv.org/abs/1607.06450.

Batson, Joshua, and Loic Royer. 2019. "Noise2Self: Blind Denoising by Self-Supervision." *International Conference on Machine Learning (ICML)*.

Brenier, Yann. 1991. "Polar Factorization and Monotone Rearrangement of Vector-Valued Functions." *Communications on Pure and Applied Mathematics* 44 (4): 375–417.

Broaddus, Coleman, Alexander Krull, Martin Weigert, Uwe Schmidt, and Gene Myers. 2020. "Removing Structured Noise with Self-Supervised Blind-Spot Networks." *IEEE International Symposium on Biomedical Imaging (ISBI)*.

Burgess, Christopher P., Irina Higgins, Arka Pal, et al. 2018. *Understanding Disentangling in Beta-VAE*. https://arxiv.org/abs/1804.03599.

Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1606.03657.

Choi, Yunjey, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. 2018. "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1711.09020.

Choi, Yunjey, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. 2020. "StarGAN V2: Diverse Image Synthesis for Multiple Domains." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1912.01865.

Chu, Casey, Andrey Zhmoginov, and Mark Sandler. 2017. *CycleGAN, a Master of Steganography*. https://arxiv.org/abs/1712.02950.

Csiszár, Imre. 1967. "Information-Type Measures of Difference of Probability Distributions and Indirect Observations." *Studia Scientiarum Mathematicarum Hungarica* 2: 299–318.

Dabov, Kostadin, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. 2007. "Image Denoising by Sparse 3-d Transform-Domain Collaborative Filtering." *IEEE Transactions on Image Processing* 16 (8): 2080–95.

Dinh, Laurent, David Krueger, and Yoshua Bengio. 2015. "NICE: Non-Linear Independent Components Estimation." *International Conference on Learning Representations (ICLR), Workshop Track*. https://arxiv.org/abs/1410.8516.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. 2017. "Density Estimation Using Real NVP." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1605.08803.

Efron, Bradley. 2011. "Tweedie's Formula and Selection Bias." *Journal of the American Statistical Association* 106 (496): 1602–14.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. 2016. *Image Style Transfer Using Convolutional Neural Networks*.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, et al. 2014. "Generative Adversarial Nets." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1406.2661.

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and

Alexander Smola. 2012. "A Kernel Two-Sample Test." *Journal of Machine Learning Research* 13: 723–73.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. "Improved Training of Wasserstein GANs." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1704.00028.

He, Junxian, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. "Lagging Inference Networks and Posterior Collapse in Variational Autoencoders." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1901.05534.

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1706.08500.

Higgins, Irina, Loïc Matthey, Arka Pal, et al. 2017. "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." *International Conference on Learning Representations (ICLR)*.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. 2020. "Denoising Diffusion Probabilistic Models." *Advances in Neural Information Processing Systems (NeurIPS)*.

Hu, Jie, Li Shen, and Gang Sun. 2018. "Squeeze-and-Excitation Networks." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1709.01507.

Huang, Xun, and Serge Belongie. 2017. "Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization." *IEEE International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/1703.06868.

Hyvarinen, Aapo. 2005. "Estimation of Non-Normalized Statistical Models by Score Matching." *Journal of Machine Learning Research* 6: 695–709.

Ioffe, Sergey, and Christian Szegedy. 2015. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1502.03167.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. "Image-to-Image Translation with Conditional Adversarial Networks." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1611.07004.

Karras, Tero, Samuli Laine, and Timo Aila. 2019. "A Style-Based Generator Architecture

for Generative Adversarial Networks." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1812.04948.

Karras, Tero, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. "Analyzing and Improving the Image Quality of StyleGAN." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1912.04958.

Kim, Kwanyoung, and Jong Chul Ye. 2021. "Noise2Score: Tweedie's Approach to Self-Supervised Image Denoising Without Clean Images." *Advances in Neural Information Processing Systems (NeurIPS)*.

Kim, Taeksoo, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. "Learning to Discover Cross-Domain Relations with Generative Adversarial Networks." *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1703.05192.

Kingma, Diederik P., and Prafulla Dhariwal. 2018. "Glow: Generative Flow with Invertible 1x1 Convolutions." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1807.03039.

Kingma, Diederik P., Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. "Improved Variational Inference with Inverse Autoregressive Flow." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1606.04934.

Kingma, Diederik P., Tim Salimans, Ben Poole, and Jonathan Ho. 2021. "Variational Diffusion Models." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/2107.00630.

Kingma, Diederik P., and Max Welling. 2014. "Auto-Encoding Variational Bayes." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1312.6114.

Krull, Alexander, Tim-Oliver Buchholz, and Florian Jug. 2019. "Noise2Void — Learning Denoising from Single Noisy Images." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kwon, Gihyun, and Jong Chul Ye. 2021. "Diagonal Attention and Style-Based GAN for Content-Style Disentanglement in Image Generation and Translation." *IEEE/CVF International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/2103.16146.

Lee, Dongwook, Junyoung Kim, Won-Jin Moon, and Jong Chul Ye. 2019. "CollaGAN :

Collaborative GAN for Missing Image Data Imputation." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1901.09764.

Lehtinen, Jaakko, Jacob Munkberg, Jon Hasselgren, et al. 2018. "Noise2Noise: Learning Image Restoration Without Clean Data." *International Conference on Machine Learning (ICML)*.

Lei, Na, Kehua Su, Li Cui, Shing-Tung Yau, and David Xianfeng Gu. 2017. *A Geometric View of Optimal Transportation and Generative Model*. https://arxiv.org/abs/1710.05488.

Li, Yijun, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. 2017. "Universal Style Transfer via Feature Transforms." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1705.08086.

Lim, Jae Hyun, and Jong Chul Ye. 2017. *Geometric GAN*. https://arxiv.org/abs/1705.02894.

Mao, Xudong, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. "Least Squares Generative Adversarial Networks." *IEEE International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/1611.04076.

McCann, Robert J. 1995. "Existence and Uniqueness of Monotone Measure-Preserving Maps." *Duke Mathematical Journal* 80 (2): 309–23.

Metz, Luke, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2017. "Unrolled Generative Adversarial Networks." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1611.02163.

Miyato, Takeru, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. "Spectral Normalization for Generative Adversarial Networks." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1802.05957.

Moran, Nick, Dan Schmidt, Yu Zhong, and Patrick Coady. 2020. "Noisier2Noise: Learning to Denoise from Unpaired Noisy Data." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mroueh, Youssef, Tom Sercu, and Vaibhava Goel. 2017. "McGan: Mean and Covariance Feature Matching GAN." *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1702.08398.

Nguyen, XuanLong, Martin J. Wainwright, and Michael I. Jordan. 2010. "Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization." *IEEE Transactions on Information Theory* 56 (11): 5847–61. https://arxiv.org/abs/

0809.0853.

Nichol, Alexander Quinn, and Prafulla Dhariwal. 2021. "Improved Denoising Diffusion Probabilistic Models." *International Conference on Machine Learning (ICML)* 139: 8162–71.

Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka. 2016. "F-GAN: Training Generative Neural Samplers Using Variational Divergence Minimization." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1606.00709.

Papamakarios, George, Theo Pavlakou, and Iain Murray. 2017. "Masked Autoregressive Flow for Density Estimation." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1705.07057.

Peyré, Gabriel, and Marco Cuturi. 2019. "Computational Optimal Transport." *Foundations and Trends in Machine Learning* 11 (5–6): 355–607. https://arxiv.org/abs/1803.00567.

Radford, Alec, Luke Metz, and Soumith Chintala. 2016. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1511.06434.

Ramani, Sathish, Thierry Blu, and Michael Unser. 2008. "Monte-Carlo SURE: A Black-Box Optimization of Regularization Parameters for General Denoising Algorithms." *IEEE Transactions on Image Processing* 17 (9): 1540–54.

Rezende, Danilo Jimenez, and Shakir Mohamed. 2015. "Variational Inference with Normalizing Flows." *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1505.05770.

Romano, Yaniv, Michael Elad, and Peyman Milanfar. 2017. "The Little Engine That Could: Regularization by Denoising (RED)." *SIAM Journal on Imaging Sciences* 10 (4): 1804–44.

Saharia, Chitwan, William Chan, Saurabh Saxena, et al. 2022. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/2205.11487.

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. "Improved Techniques for Training GANs." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1606.03498.

Salimans, Tim, and Jonathan Ho. 2022. "Progressive Distillation for Fast Sampling of Diffusion Models." *International Conference on Learning Representations (ICLR)*.

https://arxiv.org/abs/2202.00512.

Santambrogio, Filippo. 2015. *Optimal Transport for Applied Mathematicians*. Birkhäuser.

Sim, Byeongsu, Gyutaek Oh, Jeongsol Kim, Chanyong Jung, and Jong Chul Ye. 2020. "Optimal Transport Driven CycleGAN for Unsupervised Learning in Inverse Problems." *SIAM Journal on Imaging Sciences*. https://arxiv.org/abs/1909.12116.

Sohl-Dickstein, Jascha, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. "Deep Unsupervised Learning Using Nonequilibrium Thermodynamics." *International Conference on Machine Learning (ICML)* 37: 2256–65.

Soltanayev, Shakarim, and Se Young Chun. 2018. "Training Deep Learning Based Denoisers Without Ground Truth Data." *Advances in Neural Information Processing Systems (NeurIPS)*.

Song, Jiaming, Chenlin Meng, and Stefano Ermon. 2021. "Denoising Diffusion Implicit Models." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2010.02502.

Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. "Score-Based Generative Modeling Through Stochastic Differential Equations." *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2011.13456.

Sriperumbudur, Bharath K., Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. 2009. *On Integral Probability Metrics, Phi-Divergences and Binary Classification*. https://arxiv.org/abs/0901.2698.

Stein, Charles M. 1981. "Estimation of the Mean of a Multivariate Normal Distribution." *The Annals of Statistics* 9 (6): 1135–51.

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. 2016. *Instance Normalization: The Missing Ingredient for Fast Stylization*. https://arxiv.org/abs/1607.08022.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, et al. 2017. "Attention Is All You Need." *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1706.03762.

Villani, Cédric. 2003. *Topics in Optimal Transportation*. Vol. 58. Graduate Studies in Mathematics. American Mathematical Society.

Villani, Cédric. 2009. *Optimal Transport: Old and New*. Vol. 338. Grundlehren Der Mathematischen Wissenschaften. Springer.

Vincent, Pascal. 2011. "A Connection Between Score Matching and Denoising Autoencoders." *Neural Computation* 23 (7): 1661–74.

Wu, Yuxin, and Kaiming He. 2018. "Group Normalization." *European Conference on Computer Vision (ECCV)*. https://arxiv.org/abs/1803.08494.

Ye, Jong Chul. 2022. *Geometry of Deep Learning: A Signal Processing Perspective*. Mathematics in Industry. Springer. https://doi.org/10.1007/978-981-16-6046-7.

Ye, Jong Chul. 2026a. *AI618 Generative Models and Unsupervised Learning: Lecture Slides*. KAIST, Part 1 — GAN.

Ye, Jong Chul. 2026b. *AI618 Generative Models and Unsupervised Learning: Lecture Slides*. KAIST, Part 2 — VAE.

Ye, Jong Chul. 2026c. *AI618 Generative Models and Unsupervised Learning: Lecture Slides*. KAIST, Part 3 — Self-supervised Image Denoising.

Ye, Jong Chul. 2026d. *AI618 Generative Models and Unsupervised Learning: Lecture Transcript*. KAIST, lecture 03.

Ye, Jong Chul. 2026e. *AI618: Generative Models and Unsupervised Learning, Part 4: Diffusion Model Theory*. Lecture slides, KAIST Graduate School of AI.

Yoon, Jinsung, James Jordon, and Mihaela van der Schaar. 2018. "RadialGAN: Leveraging Multiple Datasets to Improve Target-Specific Predictive Models Using Generative Adversarial Networks." *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1802.06403.

Zhang, Kai, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising." *IEEE Transactions on Image Processing* 26 (7): 3142–55.

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." *IEEE International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/1703.10593.