Lecture Script

Diffusion Models

Based on Alex Alemi's blog posts:

- *A Path to the Variational Diffusion Loss* (2022)
- *KL is All You Need* (2024)

Where we're going

[0:00–0:03]

Training a diffusion model will turn out to be remarkably simple:

1. Take a clean image x
2. Corrupt it with noise: $z_t = \underbrace{\sqrt{\bar{\alpha}_t}\, x}_{\text{signal}} + \underbrace{\sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\text{noise}}$
3. Train a network to predict the noise: $\mathcal{L} = \|\epsilon - \hat{\epsilon}_\theta(z_t, t)\|^2$

Generation is the reverse:

1. Sample pure noise $z_T \sim \mathcal{N}(0, I)$
2. For $t = T, T-1, \ldots, 1$: predict noise $\hat{\epsilon} = \hat{\epsilon}_\theta(z_t, t)$, use it to denoise one step $z_t \to z_{t-1}$
3. Return $z_0 \approx x$

That's it—a denoising problem. Training: learn to remove noise. Generation: start from pure noise and apply the denoiser repeatedly. The rest of this lecture derives *why* this works, starting from the same KL recipe we used for VAEs.

Recap: VAEs

[0:03–0:10]

Last time we developed a recipe for generative modeling based on two properties of the KL divergence:

Non-negativity: $\mathrm{KL}[p; q] \geq 0$, with equality iff $p = q$.

Monotonicity: $\mathrm{KL}[p(x, z);\, q(x, z)] \geq \mathrm{KL}[p(x);\, q(x)]$. Observing more variables can only make it easier to tell two distributions apart.

The Recipe

1. Write down the **real world** P (forward process).
2. Write down the **dream world** Q (reverse process).
3. **Minimize** $\mathrm{KL}[P; Q]$.

The VAE in one slide

We designed two joint distributions and tried to align them:

- **Forward** (real world): $p(x, z) = p(x)\, p_\phi(z|x)$ — take an image, encode it
- **Reverse** (dream world): $q(x, z) = q(z)\, q_\theta(x|z)$ — sample from prior, decode

By monotonicity, minimizing the joint KL guarantees progress on the marginal KL over images: $\mathrm{KL}[p(x, z); q(x, z)] \geq \mathrm{KL}[p(x); q(x)] \geq 0$.

The joint KL splits (up to a constant) into two terms we can optimize:

Distortion (how well does the decoder reconstruct?):

$$D = \langle -\log q_\theta(x \mid z) \rangle$$

Rate (how far is the encoding from the prior?):

$$R = \langle \mathrm{KL}(p_\phi(z|x) \| q(z)) \rangle$$

Good codes are cheap (R small) but informative enough to reconstruct (D small).

Where VAEs struggle

The decoder must bridge $\mathcal{N}(0, I) \to$ complex data **in one step**. That's a huge gap—a single nonlinear map has to do all the work.

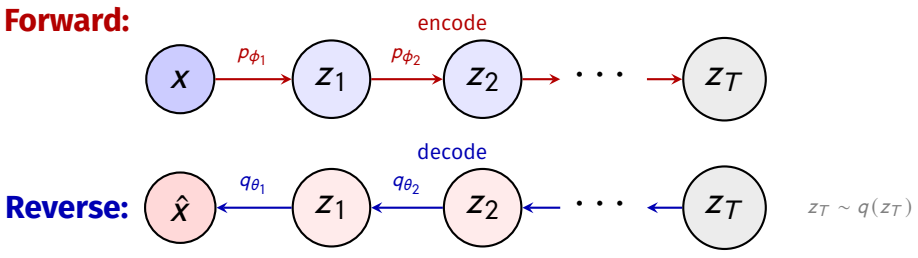
What if we used **many small steps** instead?

▹ This is the idea behind diffusion.

Diffusion and iterative refinement [0:10–0:35]

Hierarchical latent variables

Instead of one latent layer z, use a chain of T:



Same recipe applies: forward process P, reverse process Q, minimize $\mathrm{KL}[P; Q]$.

Joint KL still bounds the marginal KL on x (monotonicity). Same guarantee as before.

The problem with hierarchical VAEs

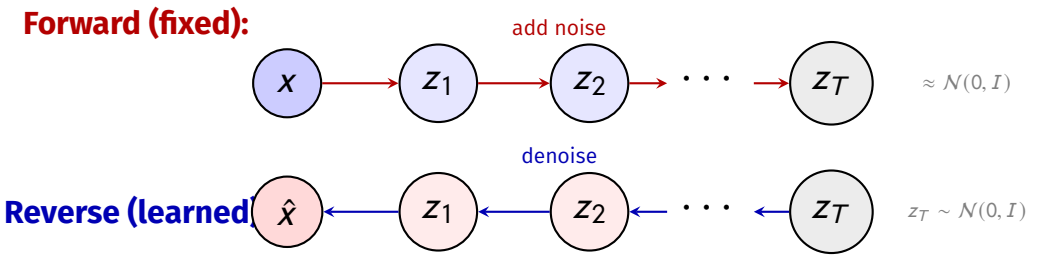
In a general hierarchical VAE, we need to learn T encoders *and* T decoders. Three costs bite:

1. **Parameters:** every layer has its own encoder and decoder to train.
2. **Training cost:** evaluating the ELBO requires running through all T layers—cost scales with depth.
3. **Forward cost:** reaching an intermediate latent z_t requires running the first t steps of the chain.

▹ Diffusion's key insight: fix the forward process.

Fix the forward process

Don't *learn* the encoder. Just add Gaussian noise:



$$p(z_t \mid z_{t-1}) = \mathcal{N}\left(z_t;\ \sqrt{1-\beta_t}\, z_{t-1},\ \beta_t\, I\right)$$

Each step shrinks the signal slightly and adds a little noise. After T steps (with a schedule that fully washes out the signal), $z_T \approx \mathcal{N}(0, I)$—pure noise.

The forward process has **zero learnable parameters**—all three problems with hierarchical VAEs vanish. Because every step is Gaussian, we also get two crucial properties that we'll exploit heavily:

Diffusion kernel (jump to any timestep). Gaussians compose. Let's see why by working out the first two steps on the board. Writing each step as $z_t = \sqrt{1-\beta_t}\, z_{t-1} + \sqrt{\beta_t}\, \epsilon_t$:

$$z_1 = \sqrt{1-\beta_1}\, z_0 + \sqrt{\beta_1}\, \epsilon_1$$
$$z_2 = \sqrt{1-\beta_2}\, z_1 + \sqrt{\beta_2}\, \epsilon_2$$

Substitute z_1 into z_2:

$$z_2 = \sqrt{(1-\beta_2)(1-\beta_1)}\, z_0 \; + \; \underbrace{\sqrt{1-\beta_2}\,\sqrt{\beta_1}\,\epsilon_1 + \sqrt{\beta_2}\,\epsilon_2}_{\text{sum of two independent Gaussians}}$$

The two noise terms are independent mean-zero Gaussians. Their sum is again Gaussian with variance $(1-\beta_2)\beta_1 + \beta_2 = 1 - (1-\beta_1)(1-\beta_2)$. So the whole thing is:

$$z_2 = \sqrt{(1-\beta_1)(1-\beta_2)}\, z_0 + \sqrt{1-(1-\beta_1)(1-\beta_2)}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

The pattern continues by induction. Defining $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$:

$$z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

A slider between data and noise. No need to run the chain step by step—we can jump directly to any z_t.

Can we reverse the process? Suppose we observe a noisy z_t and want to denoise it by one step—what is $p(z_{t-1} \mid z_t)$? By Bayes' rule:

$$p(z_{t-1} \mid z_t) = \frac{p(z_t \mid z_{t-1})\, p(z_{t-1})}{p(z_t)}$$

The numerator $p(z_t \mid z_{t-1})$ is known (it's the forward step). But $p(z_{t-1})$ and $p(z_t)$ are the **marginal distributions**—and those depend on the data distribution $p(z_0)$, which is the thing we're trying to learn. So the "naive" reverse $p(z_{t-1} \mid z_t)$ is **intractable**.

Known reverse conditionals (with z_0). But what if we peek at the clean data z_0? Then instead of the unknown marginals, we can condition on z_0 and use the diffusion kernel $p(z_{t-1} \mid z_0)$, which we just derived. By Bayes' rule:

$$p(z_{t-1} \mid z_t, z_0) = \frac{p(z_t \mid z_{t-1})\, p(z_{t-1} \mid z_0)}{p(z_t \mid z_0)}$$

All three terms on the right are **known Gaussians**: $p(z_t \mid z_{t-1})$ is the forward step, and $p(z_{t-1} \mid z_0)$ and $p(z_t \mid z_0)$ are diffusion kernels. A product of Gaussians is Gaussian, so we can read off the result:

$$p(z_{t-1} \mid z_t, z_0) = \mathcal{N}\big(z_{t-1};\ \tilde{\mu}_t(z_t, z_0),\ \tilde{\beta}_t\, I\big)$$

where

$$\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, z_0 \;+\; \frac{\sqrt{1-\beta_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, z_t, \qquad \tilde{\beta}_t = \frac{(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\,\beta_t$$

The mean is a weighted combination of z_0 and z_t (linear in both), and the variance depends only on the schedule. This is the "answer key" for each denoising step—but we only know it if we peek at z_0. Our model won't have z_0, so we'll need to estimate it.

▸ Now we derive the loss.

The loss

[0:35–0:55]

Same recipe as the VAE. Write down forward and reverse joints, minimize their KL. By monotonicity, pushing down the joint KL improves the generative model. Rewriting the forward joint using the chain rule in reverse order (conditioning each step on z_0) gives us **per-step comparisons**:

$$\mathrm{KL}[\text{joint}] \;=\; \underbrace{\mathrm{KL}[p(z_T|z_0)\|q(z_T)]}_{\text{prior matching (fixed)}} \;+\; \sum_{t=1}^{T} \underbrace{\mathrm{KL}\big[p(z_{t-1}|z_t, z_0)\;\|\;q_\theta(z_{t-1}|z_t)\big]}_{\text{per-step denoising KL}}$$

Each term compares our learned reverse step to the answer key. Now watch how this simplifies in four steps:

Step 1: Both sides are same-variance Gaussians. The answer key $p(z_{t-1} \mid z_t, z_0)$ and our model $q_\theta(z_{t-1} \mid z_t)$ both use the reverse conditional formula with variance $\tilde{\beta}_t\, I$—they only differ in whether the mean uses the true z_0

or an estimate $\hat{z}_0$. The KL between same-variance Gaussians is just the squared distance between their means:

$$\text{per-step KL} \;\propto\; \|\tilde{\mu}_t(z_t, z_0) - \tilde{\mu}_t(z_t, \hat{z}_0)\|^2 \;\propto\; \|z_0 - \hat{z}_0\|^2$$

(The second proportionality holds because $\tilde{\mu}_t$ is linear in z_0.)

Step 2: Reparameterize—predict noise instead of z_0. From the diffusion kernel, $z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$, so predicting z_0 and predicting ϵ are equivalent. Predicting noise is cleaner: $\epsilon \sim \mathcal{N}(0, I)$ is the same kind of target at every timestep. So we train $\hat{\epsilon}_\theta(z_t, t)$ and the per-step KL becomes:

$$\propto\; \|\epsilon - \hat{\epsilon}_\theta(z_t, t)\|^2$$

Step 3: Sample one t. The full loss is a sum over all t. Approximate by sampling a single random timestep:

$$\boxed{\mathcal{L} = \mathbb{E}_{t,\,x,\,\epsilon}\left[\|\epsilon - \hat{\epsilon}_\theta(z_t, t)\|^2\right]}$$

where $t \sim \text{Uniform}(1, T)$, $x \sim p(x)$, $\epsilon \sim \mathcal{N}(0, I)$, and $z_t = \sqrt{\bar{\alpha}_t}\, x + \sqrt{1-\bar{\alpha}_t}\, \epsilon$.

Add noise, predict the noise, take the squared error. That's the whole loss.

Training

For each mini-batch:

1. Sample data $x \sim p(x)$
2. Sample timestep $t \sim \text{Uniform}(1, T)$
3. Sample noise $\epsilon \sim \mathcal{N}(0, I)$
4. Compute $z_t = \sqrt{\bar{\alpha}_t}\, x + \sqrt{1-\bar{\alpha}_t}\, \epsilon$
5. Loss: $\|\epsilon - \hat{\epsilon}_\theta(z_t, t)\|^2$

One forward pass, one timestep, one noise sample. Training cost does not grow with T.

The same network sees all noise levels because we tell it t. At low t: image is slightly noisy—the model learns fine details. At high t: image is mostly noise—the model learns global structure.

Generation

Start from noise, denoise step by step:

1. Sample $z_T \sim \mathcal{N}(0, I)$
2. For $t = T, T-1, \ldots, 1$:
   - Predict noise: $\hat{\epsilon} = \hat{\epsilon}_\theta(z_t, t)$
   - Estimate the clean image: $\hat{z}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(z_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}\right)$
   - Sample z_{t-1} from the reverse conditional $p(z_{t-1} \mid z_t, \hat{z}_0)$ using $\tilde{\mu}_t(\hat{z}_0, z_t)$ and $\tilde{\beta}_t$ derived earlier (skip noise at $t = 1$)

Why iterate? The network predicts the total noise ϵ between z_0 and z_t, so in principle we could try to recover z_0 in one shot. But the prediction is an *average* over all clean images consistent with z_t—and at high noise levels, many images look alike after corruption, so that average is blurry. By stepping back to z_{t-1} (a conservative move) and re-predicting at the lower noise level, the network gets a cleaner input and a sharper estimate. Each iteration refines the picture, from coarse structure to fine detail.

Summary [0:55–1:00]

We turned unsupervised generation into supervised denoising:

1. **Fix the forward process**—just add Gaussian noise. Zero learnable parameters.
2. **The reverse needs** z_0—but we can predict the noise and recover $\hat{z}_0$.
3. **KL collapses to MSE**—teacher and student are same-variance Gaussians.

4. **Iterate to refine**—each step produces a better $\hat{z}_0$; generation goes coarse to fine.
