Generative Models
Supervised vs Unsupervised Learning

Supervised Learning

Data: \((x, y)\)

\(x\) is data, \(y\) is label

Goal: Learn a \textit{function} to map \(x \rightarrow y\)

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Cat
Supervised vs Unsupervised Learning

Supervised Learning

**Data:** (x, y)
x is data, y is label

**Goal:** Learn a *function* to map x -> y

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

Object Detection

DOG, DOG, CAT
Supervised vs Unsupervised Learning

Supervised Learning

**Data:** (x, y)
x is data, y is label

**Goal:** Learn a *function* to map x → y

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.
Supervised vs Unsupervised Learning

**Supervised Learning**

**Data:** (x, y)

x is data, y is label

**Goal:** Learn a *function* to map x -> y

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

*Image captioning*

A cat sitting on a suitcase on the floor
Supervised vs Unsupervised Learning

**Supervised Learning**

**Data:** $(x, y)$  
$x$ is data, $y$ is label

**Goal:** Learn a *function* to map $x \rightarrow y$

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

**Unsupervised Learning**

**Data:** $x$  
Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.
Supervised vs Unsupervised Learning

**Unsupervised Learning**

**Data:** x
Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.

**Clustering** (e.g. K-Means)
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x  
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Dimensionality Reduction (e.g. Principal Components Analysis)

This image from Matthias Scholz is CC0 public domain
Supervised vs Unsupervised Learning

Feature Learning (e.g. autoencoders)

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.
Unsupervised Learning

**Data:** x
Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.

Images *left* and *right* are CC0 public domain

---

Slide from Justin Johnson
Supervised vs Unsupervised Learning

Learning the distribution
e.g. sampling from it

Unsupervised Learning

Data: $x$

Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Images left and right are CC0 public domain

Slide from Justin Johnson
Supervised vs Unsupervised Learning

**Supervised Learning**

**Data:** \((x, y)\)

\(x\) is data, \(y\) is label

**Goal:** Learn a *function* to map \(x \rightarrow y\)

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

**Unsupervised Learning**

**Data:** \(x\)

Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.
Types of Generative Models

EBM: Approximate Maximum likelihood

GAN: Adversarial training

VAE: Maximize variational lower bound

Flow-based Model: Invertible transform of distributions

Diffusion Model: Gradually add Gaussian noise and then reverse

Autoregressive model: Learn conditional of each variable given past

Figure from Probabilistic Machine Learning: Advanced Topics, adapted from https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
Application of Generative Models (Image in-painting)
Application of Generative Models (As a prior)

Dream Fusion: Text-to-3D Using 2D Diffusion
Application of Generative Models (As a prior)

Dream Fusion: Text-to-3D Using 2D Diffusion
Application of Generative Models (Image generation)

Text-to-Image Synthesis on LAION 1.45B Model.

- 'A street sign that reads “Latent Diffusion”'
- 'A zombie in the style of Picasso'
- 'An image of an animal half mouse half octopus'
- 'An illustration of a slightly conscious neural network'
- 'A painting of a squirrel eating a burger'
- 'A watercolor painting of a chair that looks like an octopus'
- 'A shirt with the inscription: “I love generative models!”'
Variational Autoencoders
Variational Autoencoders

PixelRNN / PixelCNN explicitly parameterizes density function with a neural network, so we can train to maximize likelihood of training data:

$$p_w(x) = \prod_{t=1}^{T} p_w(x_t | x_1, \ldots, x_{t-1})$$

Variational Autoencoders (VAE) define an **intractable density** that we cannot explicitly compute or optimize

But we will be able to directly optimize a **lower bound** on the density
Variational Autoencoders
(Regular, non-variational) Autoencoders

Unsupervised method for learning feature vectors from raw data $x$, without any labels.

Features should extract useful information (maybe object identities, properties, scene type, etc.) that we can use for downstream tasks.

**Originally**: Linear + nonlinearity (sigmoid)
**Later**: Deep, fully-connected
**Later**: ReLU CNN
(Regular, non-variational) Autoencoders

**Problem:** How can we learn this feature transform from raw data?

Features should extract useful information (maybe object identities, properties, scene type, etc) that we can use for downstream tasks. But we can’t observe features!

- **Originally:** Linear + nonlinearity (sigmoid)
- **Later:** Deep, fully-connected
- **Later:** ReLU CNN

Slide from Justin Johnson
(Regular, non-variational) Autoencoders

**Problem:** How can we learn this feature transform from raw data?

**Idea:** Use the features to **reconstruct** the input data with a **decoder**

“Autoencoding” = encoding itself

---

Reconstructed input data \( \hat{X} \)

Decoder

Features \( Z \)

Encoder

Input data \( X \)

---

Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN (upconv)

---

Input Data

---

Slide from Justin Johnson
(Regular, non-variational) Autoencoders

**Loss:** L2 distance between input and reconstructed data.

\[ \| \hat{x} - x \|_2^2 \]

Does not use any labels! Just raw data!

---

Slide from Justin Johnson
(Regular, non-variational) Autoencoders

**Loss:** L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!

**Loss Function**

\[ \| \hat{x} - x \|_2^2 \]

Reconstructed input data \( \hat{x} \)

Decoder: 4 tconv layers
Encoder: 4 conv layers

Input data \( x \)

Features \( z \)

Slide from Justin Johnson
(Regular, non-variational) Autoencoders

**Loss:** L2 distance between input and reconstructed data.

Loss Function: \[ \| \hat{x} - x \|_2^2 \]

- **Input data**
- **Features**
- **Encoder:** 4 conv layers
- **Features need to be lower dimensional than the data**
- **Decoder:** 4 tconv layers
- **Reconstructed data**
- **Loss:** Does not use any labels! Just raw data!
(Regular, non-variational) Autoencoders

After training, **throw away decoder** and use encoder for a downstream task.

Slide from Justin Johnson
(Regular, non-variational) Autoencoders

After training, **throw away decoder** and use encoder for a downstream task

- **Input data** $\mathbf{x}$
- **Features** $\mathbf{z}$
- **Predicted Label** $\hat{y}$
- **Y**
- **Loss function** (Softmax, etc)
- **Classifier**
- **Encoder**

Fine-tune encoder jointly with classifier

Encoder can be used to initialize a **supervised** model

Train for final task (sometimes with small data)

Slide from Justin Johnson
(Regular, non-variational) Autoencoders

Autoencoders learn **latent features** for data without any labels!
Can use features to initialize a **supervised** model
Not probabilistic: No way to sample new data from learned model

Slide from Justin Johnson
Variational Autoencoders

Kingma and Welling, Auto-Encoding Variational Bayes, ICLR 2014
Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features $z$ from raw data
2. Sample from the model to generate new data
Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features $z$ from raw data
2. Sample from the model to generate new data

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

**Intuition:** $x$ is an image, $z$ is latent factors used to generate $x$: attributes, orientation, etc.
Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features $z$ from raw data
2. Sample from the model to generate new data

After training, sample new data like this:

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

**Intuition:** $x$ is an image, $z$ is latent factors used to generate $x$: attributes, orientation, etc.
Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features $z$ from raw data
2. Sample from the model to generate new data

After training, sample new data like this:

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

**Intuition:** $x$ is an image, $z$ is latent factors used to generate $x$: attributes, orientation, etc.

Assume simple prior $p(z)$, e.g. Gaussian
Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features $z$ from raw data
2. Sample from the model to generate new data

After training, sample new data like this:

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

**Intuition:** $x$ is an image, $z$ is latent factors used to generate $x$: attributes, orientation, etc.

Assume simple prior $p(z)$, e.g. Gaussian

Represent $p(x|z)$ with a neural network (Similar to decoder from autencoder)
Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample $z$ from prior $p_{\theta^*}(z)$

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

**Intuition**: $x$ is an image, $z$ is latent factors used to generate $x$: attributes, orientation, etc.

Assume simple prior $p(z)$, e.g. Gaussian

Represent $p(x \mid z)$ with a neural network (Similar to decoder from autencoder)
Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample $z$ from prior $p_{\theta^*}(z)$

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from unobserved (latent) representation $z$

How to train this model?

Basic idea: **maximize likelihood of data**

If we could observe the $z$ for each $x$, then could train a *conditional generative model* $p(x \mid z)$
Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

How to train this model?

Basic idea: **maximize likelihood of data**

We don’t observe $z$, so need to marginalize:

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p_\theta(z)dz$$
Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

How to train this model?

Basic idea: **maximize likelihood of data**

We don’t observe $z$, so need to marginalize:

$$p_\theta(x) = \int p_\theta(x, z)dz = \int \left[p_\theta(x|z)p_\theta(z)dz\right]$$

Ok, can compute this with decoder network
Variational Autoencoders

Decoder must be **probabilistic**: Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from unobserved (latent) representation $z$

How to train this model?

Basic idea: **maximize likelihood of data**

We don’t observe $z$, so need to marginalize:

$$p_{\theta}(x) = \int p_{\theta}(x, z) dz = \int p_{\theta}(x|z)p_{\theta}(z) dz$$

Ok, we assumed Gaussian prior for $z$

Sample from conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample $z$ from prior $p_{\theta^*}(z)$

Slide from Justin Johnson
Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation $z$

How to train this model?

Basic idea: maximize likelihood of data

We don’t observe $z$, so need to marginalize:

$$p_{\theta}(x) = \int p_{\theta}(x, z)dz = \int p_{\theta}(x|z)p_{\theta}(z)dz$$

Problem: Impossible to integrate over all $z$!
Variational Autoencoders

Decoder must be **probabilistic**: Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$.

Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$.

Recall $p(x, z) = p(x | z)p(z) = p(z | x)p(x)$.

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from unobserved (latent) representation z.

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes’ Rule:

$$p_\theta(x) = \frac{p_\theta(x | z)p_\theta(z)}{p_\theta(z | x)}$$
Variational Autoencoders

Decoder must be \textbf{probabilistic}:
Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from unobserved (latent) representation $z$

How to train this model?

Basic idea: \textbf{maximize likelihood of data}

Another idea: Try Bayes’ Rule:

$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

Ok, compute with decoder network
Variational Autoencoders

Decoder must be **probabilistic**: Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$.

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$.

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes’ Rule:

$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

Ok, we assumed Gaussian prior

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from unobserved (latent) representation $z$.
Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs $z$, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample $x$ from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional $p_{\theta^*}(x \mid z^{(i)})$
Sample $z$ from prior $p_{\theta^*}(z)$

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from unobserved (latent) representation $z$

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes’ Rule:

$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

**Problem**: No way to compute this!
Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional $p_{\theta^*}(x \mid z^{(i)})$
Sample z from prior $p_{\theta^*}(z)$

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation z

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes’ Rule:

$$p_{\theta}(x) = \frac{p_{\theta}(x \mid z)p_{\theta}(z)}{p_{\theta}(z \mid x)}$$

**Solution**: Train another network (encoder) that learns $q_{\phi}(z \mid x) \approx p_{\theta}(z \mid x)$
Variational Autoencoders

Decoder must be **probabilistic**: Decoder inputs \( z \), outputs mean \( \mu_{x|z} \) and (diagonal) covariance \( \Sigma_{x|z} \).

Sample \( x \) from Gaussian with mean \( \mu_{x|z} \) and (diagonal) covariance \( \Sigma_{x|z} \).

| Sample from conditional \( p_{\theta^*}(x | z^{(i)}) \) |
|----------------------------------|
| Sample \( z \) from prior \( p_{\theta^*}(z) \) |
| \( \mu_{x|z} \) | \( \Sigma_{x|z} \) |

Recall \( p(x, z) = p(x | z)p(z) = p(z | x)p(x) \)

Assume training data \( \{x^{(i)}\}_{i=1}^{N} \) is generated from unobserved (latent) representation \( z \).

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

\[
p_\theta(x) = \frac{p_\theta(x | z)p_\theta(z)}{p_\theta(z | x)} \approx \frac{p_\theta(x | z)p_\theta(z)}{q_\phi(z | x)}
\]

Use **encoder** to compute \( q_\phi(z | x) \approx p_\theta(z | x) \)

Slide from Justin Johnson
Variational Autoencoders

**Decoder network** inputs latent code $z$, gives distribution over data $x$

$$p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$$

**Encoder network** inputs data $x$, gives distribution over latent codes $z$

$$q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$

If we can ensure that $q_\phi(z \mid x) \approx p_\theta(z \mid x)$,

then we can approximate

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

**Idea:** Jointly train both encoder and decoder
Variational Autoencoders

\[ \log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} \]

Bayes’ Rule
Variational Autoencoders

\[ \log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)} \]

Multiply top and bottom by \( q_\phi(z|x) \)
Variational Autoencoders

\[
\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x \mid z)p(z)q_\phi(z \mid x)}{p_\theta(z \mid x)q_\phi(z \mid x)}
\]

\[
= \log p_\theta(x \mid z) - \log \frac{q_\phi(z \mid x)}{p(z)} + \log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}
\]

Split up using rules for logarithms
Variational Autoencoders

\[
\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)q_\phi(z \mid x)}
\]

\[
= \log p_\theta(x \mid z) - \log \frac{q_\phi(z \mid x)}{p(z)} + \log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}
\]

Split up using rules for logarithms
Variational Autoencoders

\[
\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x\mid z)p(z)q_\phi(z\mid x)}{p_\theta(z\mid x)q_\phi(z\mid x)} \\
= \log p_\theta(x \mid z) - \log \frac{q_\phi(z \mid x)}{p(z)} + \log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \\
\log p_\theta(x) = E_{z \sim q_\phi(z \mid x)}[\log p_\theta(x)] \quad \text{We can wrap in an expectation since it doesn’t depend on } z
\]
Variational Autoencoders

\[
\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x \mid z)p(z)q_\phi(z \mid x)}{p_\theta(z \mid x)q_\phi(z \mid x)}
\]

\[
= E_z[\log p_\theta(x \mid z)] - E_z \left[ \log \frac{q_\phi(z \mid x)}{p(z)} \right] + E_z \left[ \log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \right]
\]

\[
\log p_\theta(x) = E_{z \sim q_\phi(z \mid x)}[\log p_\theta(x)]
\]

We can wrap in an expectation since it doesn’t depend on \( z \)
Variational Autoencoders

\[
\begin{align*}
\log p_\theta(x) &= \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} \\
&= \log \frac{p_\theta(x \mid z)p(z)q_\phi(z \mid x)}{p_\theta(z \mid x)q_\phi(z \mid x)} \\
&= E_z[\log p_\theta(x \mid z)] - E_z \left[ \log \frac{q_\phi(z \mid x)}{p(z)} \right] + E_z \left[ \log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \right] \\
&= E_{z \sim q_\phi(z \mid x)}[\log p_\theta(x \mid z)] - D_{KL} \left( q_\phi(z \mid x), p(z) \right) + D_{KL}(q_\phi(z \mid x), p_\theta(z \mid x))
\end{align*}
\]

Data reconstruction
Variational Autoencoders

\[
\log p_{\theta}(x) = \log \frac{p_{\theta}(x \mid z)p(z)}{p_{\theta}(z \mid x)} = \log \frac{p_{\theta}(x|z)p(z)q_\phi(z|x)}{p_{\theta}(z|x)q_\phi(z|x)}
\]

\[
= E_z[\log p_{\theta}(x|z)] - E_z \left[ \log \frac{q_\phi(z|x)}{p(z)} \right] + E_z \left[ \log \frac{q_\phi(z|x)}{p_{\theta}(z|x)} \right]
\]

\[
= E_{z \sim q_\phi(z|x)}[\log p_{\theta}(x|z)] - D_{KL} \left( q_\phi(z|x), p(z) \right) + D_{KL}(q_\phi(z|x), p_{\theta}(z|x))
\]

KL divergence between prior, and samples from the encoder network
Variational Autoencoders

\[
\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x\mid z)p(z)q_\phi(z\mid x)}{p_\theta(z\mid x)q_\phi(z\mid x)}
\]

\[
= E_z[\log p_\theta(x\mid z)] - E_z \left[ \log \frac{q_\phi(z\mid x)}{p(z)} \right] + E_z \left[ \log \frac{q_\phi(z\mid x)}{p_\theta(z\mid x)} \right]
\]

\[
= E_{z \sim q_\phi(z\mid x)}[\log p_\theta(x\mid z)] - D_{KL} \left( q_\phi(z\mid x), p(z) \right) + D_{KL} (q_\phi(z\mid x), p_\theta(z\mid x))
\]

KL divergence between encoder and posterior of decoder
Variational Autoencoders

\[
\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x \mid z)p(z)q_\phi(z \mid x)}{p_\theta(z \mid x)q_\phi(z \mid x)}
\]

\[
= E_z [\log p_\theta(x \mid z)] - E_z \left[ \log \frac{q_\phi(z \mid x)}{p(z)} \right] + E_z \left[ \log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \right]
\]

\[
= E_{z \sim q_\phi(z \mid x)} [\log p_\theta(x \mid z)] - D_{KL} \left( q_\phi(z \mid x), p(z) \right) + D_{KL}(q_\phi(z \mid x), p_\theta(z \mid x))
\]

KL is >= 0, so dropping this term gives a lower bound on the data likelihood:
Variational Autoencoders

\[
\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x \mid z)p(z)q_\phi(z \mid x)}{p_\theta(z \mid x)q_\phi(z \mid x)}
\]

\[
= E_z[\log p_\theta(x \mid z)] - E_z \left[ \log \frac{q_\phi(z \mid x)}{p(z)} \right] + E_z \left[ \log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \right]
\]

\[
= E_{z \sim q_\phi(z \mid x)}[\log p_\theta(x \mid z)] - D_{KL} \left( q_\phi(z \mid x), p(z) \right) + D_{KL}(q_\phi(z \mid x), p_\theta(z \mid x))
\]

\[
\log p_\theta(x) \geq E_{z \sim q_\phi(z \mid x)}[\log p_\theta(x \mid z)] - D_{KL} \left( q_\phi(z \mid x), p(z) \right)
\]
Variational Autoencoders

Jointly train **encoder** \( q \) and **decoder** \( p \) to maximize the variational lower bound on the data likelihood. Also called **Evidence Lower Bound (ELBO)**

\[
\log p_\theta(x) \geq E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z))
\]

**Encoder Network**

\[
q_\phi(z \mid x) = \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})
\]

**Decoder Network**

\[
p_\theta(x \mid z) = \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})
\]
Example: Fully-Connected VAE

\[ x: 28\times28 \text{ image, flattened to 784-dim vector} \]
\[ z: 20\text{-dim vector} \]

**Encoder Network**

\[ q_\phi(z \mid x) = N(\mu_{z\mid x}, \Sigma_{z\mid x}) \]

\[ \mu_{z\mid x}: 20 \]
\[ \Sigma_{z\mid x}: 20 \]

\[ \text{Linear}(784\to400) \]
\[ \text{Linear}(400\to20) \]

**Decoder Network**

\[ p_\theta(x \mid z) = N(\mu_{x\mid z}, \Sigma_{x\mid z}) \]

\[ \mu_{x\mid z}: 768 \]
\[ \Sigma_{x\mid z}: 768 \]

\[ \text{Linear}(20\to400) \]
\[ \text{Linear}(400\to768) \]
\[ \text{Linear}(400\to768) \]
Variational Autoencoders

Train by maximizing the **variational lower bound**

\[ E_{z \sim q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - D_{KL} \left( q_\phi(z|x), p(z) \right) \]
Variational Autoencoders

Train by maximizing the variational lower bound

\[ E_{z \sim q(z|x)} [\log p(x|z)] - D_{KL} \left( q(z|x), p(z) \right) \]

1. Run input data through encoder to get a distribution over latent codes
Variational Autoencoders

Train by maximizing the variational lower bound

$$E_{z \sim q(x|z)}[\log p(x|z)] - D_{KL}(q(x|z), p(z))$$

1. Run input data through encoder to get a distribution over latent codes
2. Encoder output should match the prior $$p(z)$$!
Variational Autoencoders

Train by maximizing the variational lower bound

\[ E_{z \sim q_{\phi}(z|x)}[\log p_{\theta}(x|z)] - D_{KL}(q_{\phi}(z|x), p(z)) \]

1. Run input data through encoder to get a distribution over latent codes
2. Encoder output should match the prior \(p(z)\)!

\[
-D_{KL}(q_{\phi}(z|x), p(z)) = \int_{z} q_{\phi}(z|x) \log \frac{p(z)}{q_{\phi}(z|x)} dz = \int_{z} N(z; \mu_{z|x}, \Sigma_{z|x}) \log \frac{N(z; 0, I)}{N(z; \mu_{z|x}, \Sigma_{z|x})} dz \]

\[
= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log \left((\Sigma_{z|x})_{j}^{2}\right) - (\mu_{z|x})_{j}^{2} - (\Sigma_{z|x})_{j}^{2}\right)
\]

Closed form solution when \(q_{\phi}\) is diagonal Gaussian and \(p\) is unit Gaussian!
(Assume \(z\) has dimension \(J\))
Variational Autoencoders

Train by maximizing the
variational lower bound

\[ E_{z \sim q_{\phi}(z|x)} \left[ \log p_{\theta}(x|z) \right] - D_{KL} \left( q_{\phi}(z|x), p(z) \right) \]

1. Run input data through encoder to get a distribution over latent codes

2. **Encoder output should match the prior p(z)!**

3. Sample code z from encoder output
Variational Autoencoders

Train by maximizing the variational lower bound

\[ E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x), p(z)) \]

1. Run input data through encoder to get a distribution over latent codes
2. **Encoder output should match the prior \( p(z) \)!**
3. Sample code \( z \) from encoder output
4. Run sampled code through decoder to get a distribution over data samples
Variational Autoencoders

Train by maximizing the variational lower bound

\[
E_{z \sim q_{\phi}(z|x)} \left[ \log p_{\theta}(x|z) \right] - D_{KL} \left( q_{\phi}(z|x), p(z) \right)
\]

1. Run input data through encoder to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**
3. Sample code z from encoder output
4. Run sampled code through decoder to get a distribution over data samples
5. **Original input data should be likely under the distribution output from (4)!**
Variational Autoencoders

Train by maximizing the variational lower bound

\[ E_{z \sim q_{\phi}(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_{\phi}(z|x), p(z)) \]

1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior** \( p(z) \)!
3. Sample code \( z \) from encoder output
4. Run sampled code through **decoder** to get a distribution over data samples
5. **Original input data should be likely under** the distribution output from (4)!
6. Can sample a reconstruction from (4)
Variational Autoencoders: Generating Data

After training we can generate new data!

1. Sample z from prior p(z)

Latent code

Sample z from prior p(z)

Slide from Justin Johnson
Variational Autoencoders: Generating Data

After training we can generate new data!

1. Sample \( z \) from prior \( p(z) \)
2. Run sampled \( z \) through decoder to get distribution over data \( x \)

\[
x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})
\]
Variational Autoencoders: Generating Data

After training we can generate new data!

1. Sample z from prior p(z)
2. Run sampled z through decoder to get distribution over data x
3. Sample from distribution in (2) to generate data
Variational Autoencoders: Generating Data

32x32 CIFAR-10

Labeled Faces in the Wild


Slide from Justin Johnson
Variational Autoencoders

The diagonal prior on \( p(z) \) causes dimensions of \( z \) to be independent

“Disentangling factors of variation”

Kingma and Welling, Auto-Encoding Variational Bayes, ICLR 2014

Slide from Justin Johnson
Variational Autoencoders

After training we can **edit images**

1. Run input data through **encoder** to get a distribution over latent codes
Variational Autoencoders

After training we can **edit images**

1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code $z$ from encoder output

![Diagram of Variational Autoencoder](image)
Variational Autoencoders

After training we can **edit images**

1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code $z$ from encoder output
3. Modify some dimensions of sampled code
Variational Autoencoders

After training we can **edit images**

1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code $z$ from encoder output
3. Modify some dimensions of sampled code
4. Run modified $z$ through **decoder** to get a distribution over data sample
Variational Autoencoders

After training we can **edit images**

1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code \( z \) from encoder output
3. Modify some dimensions of sampled code
4. Run modified \( z \) through **decoder** to get a distribution over data samples
5. Sample new data from (4)
Variational Autoencoders

The diagonal prior on $p(z)$ causes dimensions of $z$ to be independent

“Disentangling factors of variation”

Kingma and Welling, Auto-Encoding Variational Bayes, ICLR 2014

Slide from Justin Johnson
Variational Autoencoders: Image Editing


Slide from Justin Johnson
Diffusion Models
(Markovian) Hierarchical Variational Autoencoders

\[ p(x, z_{1:T}) = p(z_T) p_\theta(x | z_1) \prod_{t=2}^{T} p_\theta(z_{t-1} | z_t) \]

\[ q_\phi(z_{1:T} | \mathbf{x}) = q_\phi(z_1 | \mathbf{x}) \prod_{t=2}^{T} q_\phi(z_t | z_{t-1}) \]
Diffusion Models

A Markovian Hierarchical Variational Autoencoder with three key restrictions

1. The latent dimension is exactly equal to the data dimension

2. The structure of the latent encoder at each timestep is not learned; it is pre-defined as a linear Gaussian model. In other words, it is a Gaussian distribution centered around the output of the previous timestep

3. The Gaussian parameters of the latent encoders vary over time in such a way that the distribution of the latent at final timestep $T$ is a standard Gaussian
Diffusion Models

\[ q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, (1 - \alpha_t)I) \]
\[ p(x_T) = \mathcal{N}(x_T; 0, I) \]
ELBO for Diffusion Models

\[
\log p(x) \geq \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_{0:T})}{q(x_{1:T} \mid x_0)} \right] \\
= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T) \prod_{t=1}^{T-1} p_\theta(x_{t-1} \mid x_t)}{\prod_{t=1}^{T} q(x_t \mid x_{t-1})} \right] \\
= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T) p_\theta(x_0 \mid x_1) \prod_{t=2}^{T} p_\theta(x_{t-1} \mid x_t)}{q(x_1 \mid x_0) \prod_{t=2}^{T} q(x_t \mid x_{t-1})} \right] \\
= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T) p_\theta(x_0 \mid x_1) \prod_{t=2}^{T} p_\theta(x_{t-1} \mid x_t)}{q(x_1 \mid x_0) \prod_{t=2}^{T} q(x_t \mid x_{t-1}, x_0)} \right] \\
= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p_\theta(x_T)p_\theta(x_0 \mid x_1)}{q(x_1 \mid x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1}, x_0)} \right] \\
= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0 \mid x_1)}{q(x_1 \mid x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)q(x_t \mid x_0)} \right]
\]
ELBO for Diffusion Models

\[
= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right]
\]

\[
= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_T|x_0)} + \log \frac{q(x_1|x_0)}{q(x_T|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right]
\]

\[
= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_T|x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right]
\]

\[
= \mathbb{E}_{q(x_1|x_0)} \left[ \log p_\theta(x_0|x_1) \right] + \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)}{q(x_T|x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right]
\]

\[
= \mathbb{E}_{q(x_1|x_0)} \left[ \log p_\theta(x_0|x_1) \right] + \mathbb{E}_{q(x_T|x_0)} \left[ \log \frac{p(x_T)}{q(x_T|x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_{t-1}|x_t, x_0)} \left[ \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right]
\]

\[
= \mathbb{E}_{q(x_1|x_0)} \left[ \log p_\theta(x_0|x_1) \right] - D_{KL}(q(x_T|x_0) \parallel p(x_T)) - \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)} \left[ D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \right]
\]

reconstruction term  \hspace{1cm} \text{prior matching term}  \hspace{1cm} \text{denoising matching term}
ELBO for Diffusion Models

\[
\begin{align*}
= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_0)q(x_T|x_0)} \right] \\
= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \frac{q(x_T|x_0)}{q(x_T|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right] \\
= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right] \\
= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log p_\theta(x_0|x_1) \right] + \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)}{q(x_T|x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right] \\
= \mathbb{E}_{q(x_1|x_0)} \left[ \log p_\theta(x_0|x_1) \right] + \mathbb{E}_{q(x_T|x_0)} \left[ \log \frac{p(x_T)}{q(x_T|x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x, x_{t-1}|x_0)} \left[ \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \right] \\
= \mathbb{E}_{q(x_1|x_0)} \left[ \log p_\theta(x_0|x_1) \right] \quad \text{reconstruction term} \\
- \mathbb{D}_{KL}(q(x_T|x_0) \| p(x_T)) \quad \text{prior matching term} \\
- \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)} \left[ \mathbb{D}_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) \right] \quad \text{denoising matching term}
\end{align*}
\]
Computing the Denoising Matching Term

\[
q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \\
= \frac{N(x_t; \sqrt{\alpha_t}x_{t-1}, (1 - \alpha_t)I)N(x_{t-1}; \sqrt{\alpha_{t-1}}x_0, (1 - \tilde{\alpha}_{t-1})I)}{N(x_t; \sqrt{\tilde{\alpha}_t}x_0, (1 - \tilde{\alpha}_t)I)} \\
= \exp \left\{ -\frac{1}{2} \left( \frac{1}{(1 - \alpha_t)(1 - \tilde{\alpha}_{t-1})} \right) \left[ x_{t-1}^2 - \frac{2\sqrt{\alpha_t(1 - \tilde{\alpha}_{t-1})x_t + \sqrt{\tilde{\alpha}_t(1 - \alpha_t)x_0}}}{1 - \tilde{\alpha}_t} x_{t-1} \right] \right\} \\
\propto N(x_{t-1}; \frac{\sqrt{\alpha_t(1 - \tilde{\alpha}_{t-1})x_t + \sqrt{\tilde{\alpha}_t(1 - \alpha_t)x_0}}}{1 - \tilde{\alpha}_t}, \frac{(1 - \alpha_t)(1 - \tilde{\alpha}_{t-1})}{1 - \tilde{\alpha}_t}I) \\
\mu_q(x_t, x_0) \\
\Sigma_q(t)
\]
Loss Function

\[
= \mathbb{E}_{q(x_1 | x_0)} [\log p_\theta(x_0 | x_1)] - D_{KL}(q(x_T | x_0) \mid \mid p(x_T)) - \sum_{t=2}^{T} \mathbb{E}_{q(x_t | x_0)} [D_{KL}(q(x_{t-1} | x_t, x_0) \mid \mid p_\theta(x_{t-1} | x_t))]
\]

- **Reconstruction term**
- **Prior matching term**
- **Denoising matching term**

\[
q(x_{t-1} | x_t, x_0) \propto \mathcal{N}(x_{t-1}; \frac{\sqrt{\alpha_t}(1 - \tilde{\alpha}_{t-1})x_t + \sqrt{\tilde{\alpha}_{t-1}}(1 - \alpha_t)x_0}{1 - \tilde{\alpha}_t}, \frac{(1 - \alpha_t)(1 - \tilde{\alpha}_{t-1})}{1 - \tilde{\alpha}_t}\mu_q(x_t, x_0), \Sigma_q(t))
\]

We will assume \( p_\theta(x_{t-1} | x_t) \) can be approximated as a Gaussian.

\[
D_{KL}(\mathcal{N}(x; \mu_x, \Sigma_x) \mid \mid \mathcal{N}(y; \mu_y, \Sigma_y)) = \frac{1}{2} \left[ \log \frac{\Sigma_y}{\Sigma_x} - d + \text{tr}(\Sigma_y^{-1}\Sigma_x) + (\mu_y - \mu_x)^T \Sigma_y^{-1}(\mu_y - \mu_x) \right]
\]

\[
\arg \min_{\theta} D_{KL}(q(x_{t-1} | x_t, x_0) \mid \mid p_\theta(x_{t-1} | x_t)) = \arg \min_{\theta} \frac{1}{2\sigma_q^2(t)} \left[ \| \mu_\theta - \mu_q \|^2 \right]
\]

\[
= \arg \min_{\theta} \frac{1}{2\sigma_q^2(t)} \frac{\tilde{\alpha}_{t-1}(1 - \alpha_t)^2}{(1 - \tilde{\alpha}_t)^2} \left[ \| \hat{x}_\theta(x_t, t) - x_0 \|^2 \right]
\]
DDPMs: Basic idea

CVPR 2022 tutorial
DDPMs: Basic idea

- **Forward process** $q$ turns images into Gaussian noise
- **Reverse process** $p$ turns noise into images
- Provided the increments of $t$ are small enough, $p_{\theta}(x_{t-1}|x_t)$ is Gaussian and we can train a neural network to estimate the mean of $x_{t-1}$ given $x_t$

DDPMs: Basic idea

\[ \epsilon \! \sim \! \epsilon(x_t, t) \] is the predicted noise component of image \( x_t \) given noise level \( t \)

- \( \epsilon \sim \mathcal{N}(0, I) \)
- Network parameters \( \theta \) are updated to reduce L2 error between actual noise \( \epsilon \) and predicted noise \( \epsilon_{\theta}(x_t, t) \)

**Algorithm 1 Training**

1: repeat
2: \( x_0 \sim q(x_0) \)
3: \( t \sim \text{Uniform}\{1, \ldots, T\} \)
4: \( \epsilon \sim \mathcal{N}(0, I) \)
5: Take gradient descent step on \[ \nabla_{\theta} \| \epsilon - \epsilon_{\theta}(x_t, t) \|^2 \]
6: until converged

**DDPMs: Basic idea**

![DDPM diagram](https://example.com/ddpm-diagram.png)

**Algorithm 1 Training**

1: repeat
2: \( x_0 \sim q(x_0) \)
3: \( t \sim \text{Uniform}([1, \ldots, T]) \)
4: \( \epsilon \sim \mathcal{N}(0, I) \)
5: Take gradient descent step on 
   \[ \nabla_\theta \| \epsilon - \epsilon_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon, t) \|^2 \]
6: until converged

**Algorithm 2 Sampling**

1: \( x_T \sim \mathcal{N}(0, I) \)
2: for \( t = T, \ldots, 1 \) do
3: \( z \sim \mathcal{N}(0, I) \) if \( t > 1 \), else \( z = 0 \)
4: \( x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z \)
5: end for
6: return \( x_0 \)

J. Ho et al. [Denoising diffusion probabilistic models](https://example.com/dpm-paper). NeurIPS 2020
Alternate viewpoint: Score-based generative modeling

• It can be shown that $\epsilon_\theta(x_t, t) \approx -\nabla_{x_t} \log q(x_t)$, where $\nabla_{x_t} \log q(x_t)$ is the score function of the (noisy) data distribution

• To sample from the original data density $q(x_0)$, we can use annealed Langevin dynamics, i.e., start by sampling from noise-perturbed versions of the data distribution and gradually reduce the amount of noise

---

https://yang-song.net/blog/2021/score/
DDPMs: Implementation

- U-Net architectures are typically used to represent $\epsilon_\theta(x_t, t)$
- Bells and whistles: residual blocks, self-attention

Time is encoded using sinusoidal positional embeddings or random Fourier features, fed into the U-Net using addition or adaptive normalization

Source: CVPR 2002 DM tutorial