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 -> y

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

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.
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.
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.

---

**Image captioning**

*A cat sitting on a suitcase on the floor*

Caption generated using [neuraltalk2](https://github.com/k tumblr/neuraltalk2)

Image is [CC0 Public domain](https://creativecommons.org/publicdomain/zero/1.0/).
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

Clustering (e.g. K-Means)

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.
Supervised vs Unsupervised Learning

Feature Learning (e.g. autoencoders)

Unsupervised Learning

**Data:** $x$

Just data, no labels!

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

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

Learning the distribution
e.g. density estimation

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

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
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 (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)**

<table>
<thead>
<tr>
<th>Textual Description</th>
<th>Image Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'A street sign that reads “Latent Diffusion”'</td>
<td><img src="image1.png" alt="Image of a street sign" /></td>
</tr>
<tr>
<td>'A zombie in the style of Picasso'</td>
<td><img src="image2.png" alt="Image of a zombie" /></td>
</tr>
<tr>
<td>'An image of an animal half mouse half octopus'</td>
<td><img src="image3.png" alt="Image of a hybrid animal" /></td>
</tr>
<tr>
<td>'An illustration of a slightly conscious neural network'</td>
<td><img src="image4.png" alt="Image of a neural network illustration" /></td>
</tr>
<tr>
<td>'A painting of a squirrel eating a burger'</td>
<td><img src="image5.png" alt="Image of a squirrel eating" /></td>
</tr>
<tr>
<td>'A watercolor painting of a chair that looks like an octopus'</td>
<td><img src="image6.png" alt="Image of a watercolor octopus chair" /></td>
</tr>
<tr>
<td>'A shirt with the inscription: “I love generative models!”'</td>
<td><img src="image7.png" alt="Image of a generative models shirt" /></td>
</tr>
</tbody>
</table>
Variational Autoencoders
Variational Autoencoders

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

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

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

Slide from Justin Johnson
(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

![Diagram showing input data, encoder, and features]

---

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
(Regular, non-variational) Autoencoders

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

Does not use any labels! Just raw data!

Loss Function

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

- **Encoder**
- **Decoder**
- **Features**
- **Input data**

Input Data
(Regular, non-variational) Autoencoders

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

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

- **Input data**
- **Features**
- **Reconstructed input data**

- **Encoder**: 4 conv layers
- **Decoder**: 4 tconv layers

Does not use any labels! Just raw data!
(Regular, non-variational) Autoencoders

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

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

- **Input data**: Does not use any labels! Just raw data!
- **Features**: Features need to be lower dimensional than the data
- **Reconstructed input data**: Decoder: 4 tconv layers
- **Encoder**: 4 conv layers

- **Loss Function**: \(\|\hat{x} - x\|_2^2\)
- **Features**
- **Input data**
- **Decoder**
- **Encoder**

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

After training, **throw away decoder** and use encoder for a downstream task.
(Regular, non-variational) Autoencoders

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

Input data $\mathbf{x}$

Features $\mathbf{z}$

Loss function (Softmax, etc)

Predicted Label $\hat{y}$

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
Variational Autoencoders

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

Slide from Justin Johnson
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:

\[ p_{\theta^*}(x \mid z^{(i)}) \]

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.

Slide from Justin Johnson
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:

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

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

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

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 $\Sigma_{x|z}$

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 $\Sigma_{x|z}$

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 $p(x|z)$

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:

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

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:

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

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

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:

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

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 $\Sigma_{x|z}$

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:

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

Ok, we assumed Gaussian prior

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 $\Sigma_{x|z}$

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes’ Rule:

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

Problem: No way to compute this!

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 $z$
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 $p_{\theta^*}(x \mid z^{(i)})$
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:

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

Use **encoder** to compute $q_{\phi}(z \mid x) \approx p_{\theta}(z \mid x)$
Variational Autoencoders

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

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

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

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

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

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

Multiply top and bottom by $q_\phi(z \mid x)$
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

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

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

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

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

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

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

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

Data reconstruction
Variational Autoencoders

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

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

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

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

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

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

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

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

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

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

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

**Decoder Network**

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

x: 28x28 image, flattened to 784-dim vector
z: 20-dim vector

Encoder Network

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

- \(\mu_{z|x}: 20\)
- \(\Sigma_{z|x}: 20\)
- Linear(784->400)
- Linear(400->20)
- Linear(400->20)
- x: 784

Decoder Network

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

- \(\mu_{x|z}: 768\)
- \(\Sigma_{x|z}: 768\)
- Linear(400->768)
- Linear(400->768)
- Linear(20->400)
- z: 20
Variational Autoencoders

Train by maximizing the variational lower bound

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

Train by maximizing the variational lower bound

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

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

Train by maximizing the **variational lower bound**

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

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

\[
-D_{KL} \left( q_\phi(z|x), p(z) \right) = \int z q_\phi(z|x) \log \frac{p(z)}{q_\phi(z|x)} dz
\]

\[
= \int z N(z; \mu_{z|x}, \Sigma_{z|x}) \log \frac{N(z; 0, I)}{N(z; \mu_{z|x}, \Sigma_{z|x})} dz
\]

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

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

Slide from Justin Johnson
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
Variational Autoencoders

Train by maximizing the variational lower bound

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

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

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

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 \( p(z) \)
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 \( x \)
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

$\hat{X}$

Sample $x$ from

$x | z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

$\mu_{x|z}$

$\Sigma_{x|z}$

$Z$

Sample $z$ from prior $p(z)$

Decoder

Slide from Justin Johnson
Variational Autoencoders: Generating Data

32x32 CIFAR-10

Labeled Faces in the Wild

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
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
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

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
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

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

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

\[ q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1 - \alpha_t)I) \]

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

\[
\begin{align*}
&= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0 \mid x_1)}{q(x_1 \mid x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_0, x_t) q(x_t \mid x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0 \mid x_1)}{q(x_1 \mid x_0)} + \log \frac{q(x_T \mid x_0)}{q(x_T \mid x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T)p_\theta(x_0 \mid x_1)}{q(x_1 \mid x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log p_\theta(x_0 \mid x_1) \right] + \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p(x_T)}{q(x_T \mid x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_{1:T} \mid x_0)} \left[ \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} \right] \\
&= \mathbb{E}_{q(x_1 \mid x_0)} \left[ \log p_\theta(x_0 \mid x_1) \right] + \mathbb{E}_{q(x_T \mid x_0)} \left[ \log \frac{p(x_T)}{q(x_T \mid x_0)} \right] + \sum_{t=2}^{T} \mathbb{E}_{q(x_1, x_{t-1} \mid x_0)} \left[ \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} \right] \\
&= \left( \sum_{t=2}^{T} \mathbb{E}_{q(x_t \mid x_0)} \left[ \text{KL}(q(x_T \mid x_0) \mid \mid p(x_T)) \right] - \sum_{t=2}^{T} \mathbb{E}_{q(x_t \mid x_0)} \left[ \text{KL}(q(x_{t-1} \mid x_t, x_0) \mid \mid p_\theta(x_{t-1} \mid x_t)) \right] \right)
\end{align*}
\]

reconstruction term  prior matching term  denoising matching term
ELBO for Diffusion Models

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

reconstruction term

prior matching term
denoising matching term
Computing the Denoising Matching Term

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

reconstruction term
prior matching term
denoising matching term
Loss Function

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

reconstruction term  prior matching term  denoising matching term

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

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

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

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

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

Unconditional CIFAR10 sample generation

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

**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

- \( \epsilon_{\theta}(x_t, t) \) is the predicted noise component of image \( x_t \) given noise level \( t \)
- Network parameters \( \theta \) are updated to reduce L2 error between actual noise \( \epsilon \) and predicted noise \( \epsilon_{\theta}(x_t, t) \)

DDPMs: Basic idea

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 \( x_0 \)

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.

**Algorithm 1** Annealed Langevin dynamics.

**Require:** $\{\sigma_i\}_{i=1}^L, \epsilon, T$.
1. Initialize $\bar{x}_0$
2. for $i \leftarrow 1$ to $L$ do
   3. $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_i^2$ $\triangleright \alpha_i$ is the step size.
4. for $t \leftarrow 1$ to $T$ do
   5. Draw $z_t \sim \mathcal{N}(0, I)$
   6. $\bar{x}_t \leftarrow \bar{x}_{t-1} + \alpha_i \cdot s_{\theta}(\bar{x}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \cdot z_t$
5. end for
6. $x_0 \leftarrow \bar{x}_T$
7. end for
8. return $\bar{x}_T$


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