DIFFUSION MODELS, WHAT IS THAT ALL ABOUT

IS IT GOOD? OR IS IT WACK?
Diffusion models are really good at learning conditional distributions.

\[ p(x \mid y) \]
Use Case: Class-Conditioned Generation $p(\text{image} \mid \text{class\_label})$

source: Image Super-Resolution via Iterative Refinement
Use Case: Text-to-Image Generation

“a painting of a fox sitting in a field at sunrise in the style of Claude Monet”

source: ImageN, StableDiffusion, Dall-E 2.0
Use Case: Super Resolution

Class ID = 213
"Irish Setter"

Model 1

32×32

Model 2

64×64

Model 3

256×256

$p(image \mid low_res)$

source: Cascaded Diffusion Models
Recap: Generative Modeling

Recall the goal of generative modeling - learning a model of a distribution from which we can generate new samples.

Given $\mathbf{x} \sim p(\mathbf{x})$ we might want to learn $p_\theta(\mathbf{x}) \approx p(\mathbf{x})$ (modeling)

Then, we can generate new samples $\mathbf{x}^* \sim p_\theta(\mathbf{x})$ (generation)

Why is this useful?
Given $\mathbf{x} \sim p(\mathbf{x})$
Generative Modeling: Themes

What are some common themes of generative modeling?

- We want to learn some **complex** distribution \( p_\theta(x) \approx p(x) \)
- But we only have access to some **simple** distributions (such as Gaussians)
Generative Modeling: Themes

What are some common themes of generative modeling?

- We want to learn some **complex** distribution \( p_\theta(x) \approx p(x) \)
- But we only have access to some **simple** distributions (such as Gaussians)

**Idea:** Let’s learn a complex function (aka a neural network) to transform a simple distribution sample into a complex one!

- Gaussian Sample \( \approx\) (neural net) \(\Rightarrow\) Data Sample
Generative Modeling: Themes

Idea: Let’s learn a complex function (aka a neural network) to transform a simple distribution sample into a complex one!

- Gaussian Sample =>(neural net)=> Data Sample

You have seen this before in:

- VAEs
- GANs
- and now, Diffusion Models!
An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.
An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.
Diffusion Models: a TLDR

An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.
An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.
Diffusion Models: a TLDR

An observation: adding steady amounts of Gaussian noise eventually corrupts an image into something indistinguishable from a random Gaussian sample.

- Diffusion models simply learn to **reverse** this procedure over many timesteps
Recap: Variational Autoencoders 🎉

Visually, we often see a VAE as:

![VAE diagram]

How do we perform backpropagation through samples?
Recap: Reparameterization Trick

For $x \sim \mathcal{N}(x | \mu, \sigma^2)$, $\epsilon$, where $\epsilon \sim \mathcal{N}(x | 0, 1)$.
Recap: Variational Autoencoders

Visually, we often see a VAE as:

```
\[ x \rightarrow q(z|x) \rightarrow z \rightarrow p(x|z) \rightarrow x' \]
```

Or as:
Recap: Variational Autoencoders

Visually, we often see a VAE as:

Or as:
Recap: Variational Autoencoders

Visually, we often see a VAE as:

![VAE Diagram](image)

Or as:

![VAE Diagram](image)

...but what's the intuition behind what is learned?
Generative Modeling with Latent Variables

Given $\mathbf{x} \sim p(\mathbf{x})$ we might want to learn $p_\theta(\mathbf{x}) \approx p(\mathbf{x})$ (modeling)

What if we assume latent variables $\mathbf{z}$ exist?
Elon has an idea…
Elon has an idea…

yo…
Elon has an idea...

yo...
what if our *latents*...
Elon has an idea…

yo…
what if our *latents*…
had *latents*…
Hierarchical VAEs

Generalize VAEs by enabling a hierarchy of latents $z = z_1, \ldots, z_T$

This is essentially learning a bunch of stacked VAEs

$p(x|z)$

$q(z|x)$
Hierarchical VAEs

Generalize VAEs by enabling a hierarchy of latents $z = z_1, \ldots, z_T$

This is essentially learning a bunch of stacked VAEs

Disclaimer: Elon did not actually come up with this idea.
Hierarchical VAEs

Let’s think like a caveman…

- $p(x|z_1)$
- $p(z_1|z_2)$
- $p(z_{T-1}|z_T)$

- $z_1$: shadows
- $z_2$: 3d objects
- $z_T$: abstract vector
- $\ldots$
- $z_T$: really abstract vector

- $q(z_1|x)$
- $q(z_2|z_1)$
- $q(z_T|z_{T-1})$
Hierarchical VAEs

Question:
- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

...what if we assume all latent dimensions are the same?
Hierarchical VAEs

Question:
- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?
  
  ...what if we assume all latent dimensions are the same?
Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

...what if we assume all dimensions are the same?
Hierarchical VAEs

Question:
- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

…what if we assume all dimensions are the same?
Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

…what if we assume all dimensions are the same?
Hierarchical VAEs

Question:

- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

...what if we assume all dimensions are the same?
...what if we assume all encoder transitions are known Gaussians centered around their previous input?
Let’s take a look at one encoding

\[ p(x_0|x_1) \quad p(x_1|x_2) \quad p(x_{T-1}|x_T) \]

\[ q(x_1|x_0) = \mathcal{N}(x_1|x_0, \sigma_1^2I) \]

\[ x_1 \sim q(x_1|x_0) \quad \mathbf{x}_0 \quad + \quad \mathbf{\epsilon}_0 \]

reparam. trick!
Let's take a look at one encoding

$$p(x_0|x_1) \quad p(x_1|x_2) \quad p(x_{T-1}|x_T)$$

$$q(x_1|x_0) \quad q(x_2|x_1) \quad q(x_T|x_{T-1})$$

$$q(x_2|x_1) = \mathcal{N}(x_2|x_1, \sigma_2^2I)$$

$$x_2 \sim q(x_2|x_1) \quad x_1 \quad \epsilon_0 + \sigma_2^*$$

reparam. trick!
Let’s take a look at one encoding

\[ p(x_0|x_1) \quad p(x_1|x_2) \quad p(x_{T-1}|x_T) \]

\[ q(x_1|x_0) \quad q(x_2|x_1) \quad q(x_T|x_{T-1}) \]
Let’s take a look at one encoding

\[ p(x_0|x_1) \quad p(x_1|x_2) \quad p(x_{T-1}|x_T) \]

\[ q(x_1|x_0) \quad q(x_2|x_1) \quad q(x_T|x_{T-1}) \]

\[ x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_T \]

\[ x_2 \sim q(x_2|x_1) \quad x_1 \quad \epsilon_0 \]

\[ \sigma_2^* \]

reparam. trick!
Let's take a look at one encoding

\[ p(x_0 | x_1) \quad p(x_1 | x_2) \quad p(x_{T-1} | x_T) \]

\[ q(x_1 | x_0) \quad q(x_2 | x_1) \quad q(x_T | x_{T-1}) \]

Aggregate into 1 sample!

reparam. trick!
Let's take a look at one encoding

\[
p(x_0|x_1) \quad p(x_1|x_2) \quad p(x_{T-1}|x_T)
\]

\[
x_0 \quad x_1 \quad x_2 \quad \ldots \quad x_T
\]

\[
q(x_1|x_0) \quad q(x_2|x_1) \quad q(x_T|x_{T-1})
\]

where,

\[
\alpha_2 = \sqrt{\sigma_1^2 + \sigma_2^2}
\]

and,

\[
q(x_2|x_0) = \mathcal{N}(x_2|x_0, \alpha_2^2)
\]

Aggregate into 1 sample!

reparam. trick!
Individual (known) Gaussians!
\( q(x_t|0) \) is a Gaussian, for arbitrary \( t \)!

\[ q(x_t|0) = \mathcal{N}(x_t|x_0, \alpha_t^2 I), \]  where \( \alpha_0, \alpha_1, \ldots \alpha_T \) are all known/fixed.
**Hierarchical VAEs**

**Question:**
- In a VAE we learn two networks: an encoder and a decoder.
- How many do we need to learn for a Hierarchical VAE?

...what if we assume **all** dimensions are the same?
...what if we assume all encoder transitions are known Gaussians centered around their previous input?

...then we can aggregate and simplify the distribution of each intermediate “latent”!
**Diffusion Models**

It turns out, that this is exactly what a diffusion model is!

- A Hierarchical VAE with these assumptions:

  ...what if we assume all dimensions are the same?
  ...what if we assume all encoder transitions are known Gaussians centered around their previous input?
Diffusion Models

A diffusion model is implemented as a single neural network (the decoder)

Quick Quiz: The dataset we are given is only clean images — but the decoder takes in as input. How do we get our data to train on? How do we optimize our decoder? How do we know we are outputting good results?
Optimization?

We want to learn a denoising decoder:

- \( x_t \) is a Gaussian, for arbitrary \( \theta \), where \( \mu_{\text{dec}}, \sigma_{\text{dec}} \) are all known/fixed.

- \( \hat{x}_{t-1} = \mu_{\text{dec}} + \sigma_{\text{dec}} \ast \epsilon \) (reparam. trick!)

\( \epsilon \sim \mathcal{N}(0, I) \)

But what is the form of \( x_{t-1} \)?

- \( p(x_0|x_1) \) \( p(x_{t-1}|x_t) \) \( p(x_t|x_{t+1}) \) \( p(x_{T-1}|x_T) \)

\( q(x_{t-1}|x_0) \) \( q(x_t|x_0) \) \( q(x_{t+1}|x_0) \) \( q(x_T|x_0) \)

...can we formulate this as supervised learning?
Optimization?

We want to learn a denoising decoder:

\[
\hat{x}_{t-1} = \mu_{\text{dec}} + \sigma_{\text{dec}} \ast \epsilon
\]

But what is the form of \( x_{t-1} \)?

Recall that:

is a Gaussian, for arbitrary \( \cdot \)!

Decoder NN reparam. trick!

\( \epsilon \sim \mathcal{N}(0, I) \)
Optimization?

We want to learn a denoising decoder:

\[ x_t \xrightarrow{\text{Decoder NN}} \mu_{\text{dec}} + \sigma_{\text{dec}} \times \epsilon \]

But what is the form of \( x_{t-1} \)?

Recall that:

\[
q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}|x_0, \alpha^2_{t-1}) \\
\therefore x_{t-1} = x_0 + \alpha_{t-1} \times \epsilon
\]

Do we really need to predict \( \sigma_{\text{dec}} \)?

What is the ground truth signal for \( \mu_{\text{dec}} \)?

reparam. trick!

\( \epsilon \sim \mathcal{N}(0, I) \)
Optimization?

We want to learn a denoising decoder:

\[ x_t \xrightarrow{\text{Decoder NN}} \mu_{\text{dec}} \xrightarrow{} x_{t-1} \]

\[ \hat{x}_{t-1} = \hat{x}_0 + \alpha_{t-1} \cdot \epsilon \]

reparam. trick!
\[ \epsilon \sim \mathcal{N}(0, I) \]

But what is the form of \( x_{t-1} \)?

Recall that:

\[ q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}|x_0, \alpha_{t-1}^2 I) \]

\[ \therefore x_{t-1} = x_0 + \alpha_{t-1} \cdot \epsilon \]

reparam. trick!
\[ \epsilon \sim \mathcal{N}(0, I) \]

Do we really need to predict \( \sigma_{\text{dec}} \)? What is the ground truth signal for \( \mu_{\text{dec}} \)?
Optimization?

We want to learn a denoising decoder:

\[ \hat{x}_{t-1} = \hat{x}_0 + \alpha_{t-1} \cdot \epsilon \]

But what is the form of \( x_{t-1} \) ?

Recall that:

\[ q(x_{t-1}|x_0) = \mathcal{N}(x_{t-1}|x_0, \alpha_{t-1}^2 I) \]

\[ \therefore x_{t-1} = x_0 + \alpha_{t-1} \cdot \epsilon \]

Do we really need to predict \( \sigma_{dec} \)?

What is the ground truth signal for \( \mu_{dec} \)?
Optimization?

We want to learn a denoising decoder:

So in the end, a diffusion model is simply one Neural Network that predicts a clean image \( \hat{x}_0 \) from arbitrary noisified image \( x_t \).
A Diffusion Model is:

- One NN that predicts a clean image from a noisy version of the image
A Diffusion Model is:

- One NN that predicts a clean image from a noisy version of the image
Diffusion Models: A Summary

How do we perform sampling?

Decoder NN
Sampling

\[ \hat{x}_\theta(x_{t+1}, t + 1) \]

[Diagram with nodes and arrows]

- \( x_0 \)
- \( \ldots \)
- \( x_{t-1} \)
- \( x_t \)
- \( x_{t+1} \)
- \( \ldots \)
- \( x_T \)
Sampling

\[ \hat{x}_\theta(x_{t+1}, t + 1) \]

\[ q(x_t | x_0) \]
Sampling

\[ \hat{x}_\theta(x_{t+1}, t + 1) \]

\[ p(x_t|x_{t+1}) \]

\[ q(x_t|x_0) \]
Sampling

\[ \hat{x}_\theta(x_t, t) \quad \text{and} \quad p(x_t|x_{t+1}) \]
Sampling

\[ \hat{x}_\theta(x_t, t) \]

\[ p(x_t | x_{t+1}) \]

\[ q(x_{t-1} | x_0) \]
Sampling

\[ x_0 \rightarrow \ldots \rightarrow x_{t-1} \rightarrow x_t \rightarrow x_{t+1} \rightarrow \ldots \rightarrow x_T \]

\[ p(x_{t-1}|x_t) \quad p(x_t|x_{t+1}) \nabla \hat{x}_\theta(x_t, t) \]

\[ q(x_{t-1}|x_0) \]
Sampling

\[ p(x_0|x_1) \rightarrow x_0 \rightarrow \ldots \rightarrow x_{t-1} \rightarrow p(x_{t-1}|x_t) \rightarrow x_t \rightarrow p(x_t|x_{t+1}) \rightarrow x_{t+1} \rightarrow \ldots \rightarrow p(x_{T-1}|x_T) \rightarrow x_T \]
Algorithm 1 Training

1: repeat
2: \( \mathbf{x}_0 \sim q(\mathbf{x}_0) \)
3: \( t \sim \text{Uniform}(1, \ldots, T) \)
4: \( \epsilon \sim \mathcal{N}(0, \mathbf{I}) \)
5: Take gradient descent step on \( \nabla_{\theta} \| \mathbf{x}_0 - \hat{x}_\theta(\mathbf{x}_0 + \alpha_t \epsilon, t) \|^2 \)
6: until converged

Algorithm 2 Sampling

1: \( \mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I}) \)
2: for \( t = T, \ldots, 1 \):
3: \( \epsilon \sim \mathcal{N}(0, \mathbf{I}) \) if \( t > 1 \), else \( \epsilon = 0 \)
4: \( \mathbf{x}_{t-1} = \hat{x}_\theta(\mathbf{x}_t, t) + \alpha_{t-1} \epsilon \)
5: end for
6: return \( \mathbf{x}_0 \)
Examples!

Celeb-A

CIFAR-10

source: Generative Modeling by Estimating Gradients of the Data Distribution
Examples!

source: Generative Modeling by Estimating Gradients of the Data Distribution
Three Different Interpretations

It turns out, training a VDM can be done using three different interpretations:

- Predicting original image 📸 (we just did this)

- Predicting noise 🔊 (coming up!)

- Predicting score function 💯 (coming up!)
VDM as a Noise Predictor

Recall that our objective is to predict $\hat{x}_\theta(x_t, t) \approx x_0$
What does it mean intuitively?

For arbitrary $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)$, we can rewrite it as $\mathbf{x}_t = \mathbf{x}_0 + \alpha_t \mathbf{\epsilon}_0$

Predicting $\mathbf{x}_0$ determines $\mathbf{\epsilon}_0$ and vice-versa, since they sum to the same thing!
Score Functions

What are score functions?

\[ \nabla_x \log p(x) \]

Intuitively, they describe how to move in data space to improve the (log) likelihood.
Tweedie’s Formula

Mathematically, for a Gaussian variable \( z \sim \mathcal{N}(z; \mu_z, \Sigma_z) \), Tweedie’s formula states:

\[
\mathbb{E}[\mu_z | z] = z + \Sigma_z \nabla_z \log p(z)
\]

Then, since we have previously shown that:

\[
q(x_t | x_0) = \mathcal{N}(x_t; x_0, \alpha_t^2 I)
\]

By Tweedie’s Formula, we derive:

\[
\mathbb{E}[\mu_{x_t} | x_t] = x_t + \alpha_t^2 \nabla_x x_t \log p(x_t)
\]

The best estimate for the true mean is \( \mu_{x_t} = x_0 \)

\[
x_0 = x_t + \alpha_t^2 \nabla x_t \log p(x_t)
\]
Tweedie’s Formula

There exists a mathematical formula that states that:

\[ x_0 \approx x_t + \alpha_t^2 \nabla x_t \log p(x_t) \]

Due to the fact that the distribution is Gaussian:

\[ q(x_t|x_0) = \mathcal{N}(x_t \mid x_0, \alpha_t^2 I) \]
VDM as a Score Predictor

Recall that our objective is to predict $\hat{x}_\theta(x_t, t) \approx x_0$
Score 💯 and Noise 🎧?

There is a relationship between the score and the noise, which we can derive by equating Tweedie’s formula with the Reparameterization Trick.

\[
\begin{align*}
x_0 &= x_t + \alpha_t^2 \nabla \log p(x_t) = x_t - \alpha_t \epsilon_0 \\
\therefore \quad \alpha_t^2 \nabla \log p(x_t) &= -\alpha_t \epsilon_0 \\
\nabla \log p(x_t) &= -\frac{1}{\alpha_t} \epsilon_0
\end{align*}
\]

Intuitively, the direction to move in data space towards a natural image is the negative noise term that was added.
Three Different Interpretations

It turns out, training a VDM can be implemented as a neural net that:

- Predisets original image: $\hat{x}_{\theta}(x_t, t) \approx x_0$

- Predicts noise epsilon: $\hat{\epsilon}_{\theta}(x_t, t) \approx \epsilon_0$

- Predicts score function: $s_{\theta}(x_t, t) \approx \nabla_{x_t} \log p(x_t)$
A Summary

We have learned that a diffusion model is simply one neural network that predicts a clean image from a noisy image.

Objective: \[ \arg \min_{\theta} \| x_0 - \hat{x}_\theta(x_t, t) \|^2 \]

Sampling: