Diffusion Model

Computer Vision - Project

Clément, Grégoire, Nathan

January 14, 2025

What is Diffusion ?

Diffusion is the state-of-the-art generative process to generate images for creating new images near the original one. It also allows to generate images from text.

First generation: Denoising Diffusion Probabilistic Models

General Idea

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

General Idea

Consider the set of hand-written digits D. Can you give a probability distribution q such that $x \sim q(x)$?

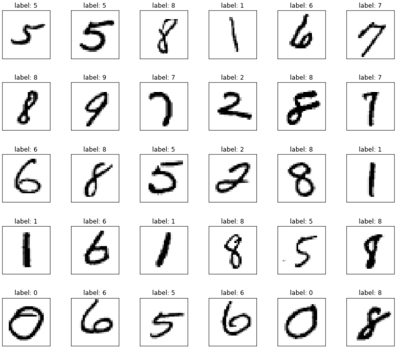
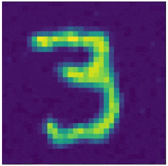


Figure 1: Source: ludwig.ai

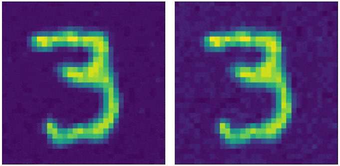
General Idea

Consider the set of hand-written digits D. It is hard to find q such that $x \sim q(x)$, we need a clever way to sample hand-written digits. Consider the following process:



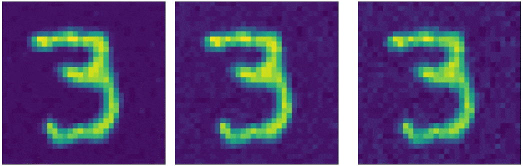
General Idea

Consider the set of hand-written digits D. It is hard to find q such that $x \sim q(x)$, we need a clever way to sample hand-written digits. Consider the following process:



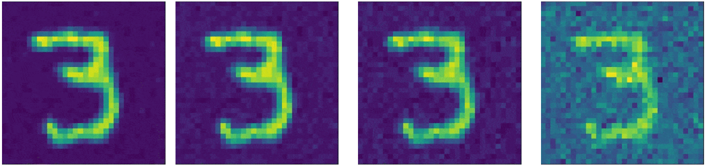
General Idea

Consider the set of hand-written digits D. It is hard to find q such that $x \sim q(x)$, we need a clever way to sample hand-written digits. Consider the following process:



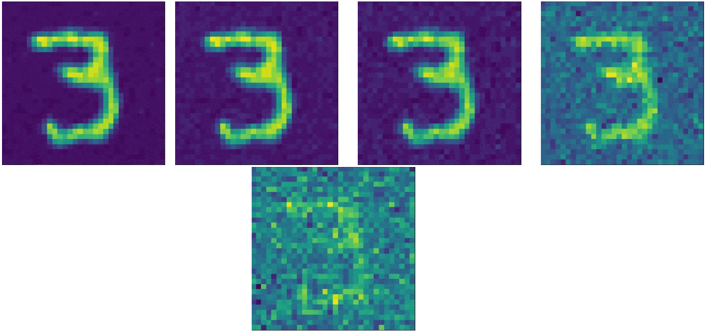
General Idea

Consider the set of hand-written digits D. It is hard to find q such that $x \sim q(x)$, we need a clever way to sample hand-written digits. Consider the following process:



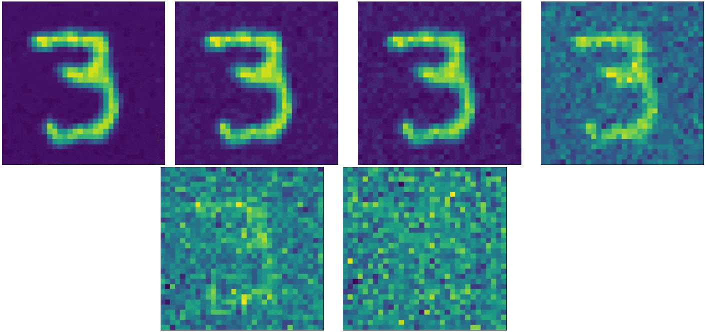
General Idea

Consider the set of hand-written digits D. It is hard to find q such that $x \sim q(x)$, we need a clever way to sample hand-written digits. Consider the following process:



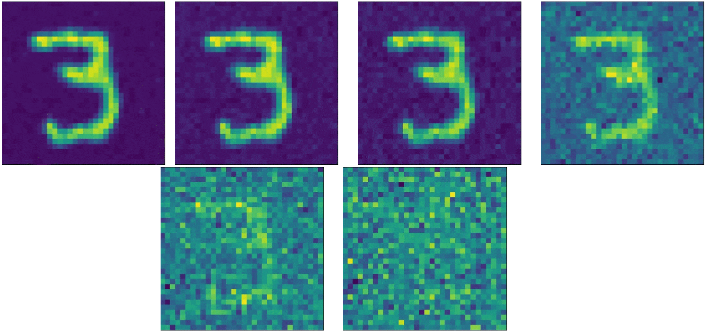
General Idea

Consider the set of hand-written digits D. It is hard to find q such that $x \sim q(x)$, we need a clever way to sample hand-written digits. Consider the following process:



General Idea

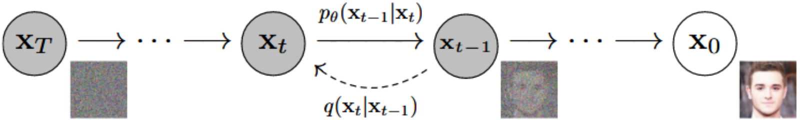
Consider the set of hand-written digits D. It is hard to find q such that $x \sim q(x)$, we need a clever way to sample hand-written digits. Consider the following process:



Formally: $q(x_{t+1} \mid x_t) := \mathcal{N}(x_{t+1}; \sqrt{1-\beta_t}x_t, \beta_t I)$ for some schedule $(\beta_t)_t$. Can we learn to reverse this process ?

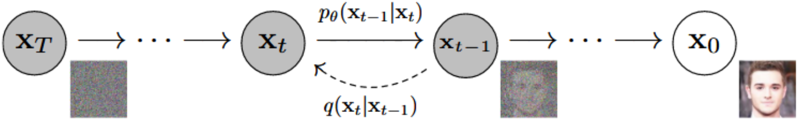
What we want to learn

Given a noisy image x_t, we train a model to predict x_{t-1}.



What we want to learn

Given a noisy image x_t, we train a model to predict x_{t-1}.



- Given a noisy image x_t and t, we sample according to $p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$.

Decreasing training data generation cost

Given a data image x_0, computing x_t takes t samplings on q. But a simple trick allows us to do it in only one step.

Remember that $q(x_{t+1} \mid x_t) := \mathcal{N}(x_{t+1}; \sqrt{1-\beta_t}x_t, \beta_t I)$. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha_t} = \prod_{i=1}^{t} \alpha_i$.

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\epsilon_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)}\bar{\epsilon_t} \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon_t}
\end{aligned}
$$

Let $G_1 \sim \mathcal{N}(0, \sigma_1^2 I)$, $G_2 \sim \mathcal{N}(0, \sigma_2^2 I)$, the sum of them gives $g_2 \sim \mathcal{N}(0, (\sigma_1^2 + \sigma_2^2)I)$.

Decreasing training data generation cost

Given a data image x_0, computing x_t takes t samplings on q. But a simple trick allows us to do it in only one step.

We have $x_t = \sqrt{\bar{\alpha_t}} x_0 + \sqrt{1 - \bar{\alpha_t}} \epsilon$.

Training

For now, our model is learning μ and Σ, i.e. we sample according to

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

They fixed fixing Σ_θ to a constant to simplify computations. So,

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t I)$$

Training

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t I)$$

Using negative log likelihood, approximations and computations, we want to minimize:

$$E_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2 \right]$$

where $\tilde{\mu}$ is the optimal mean that depends on x_0.

Training

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t I)$$

Using negative log likelihood, approximations and computations, we want to minimize:

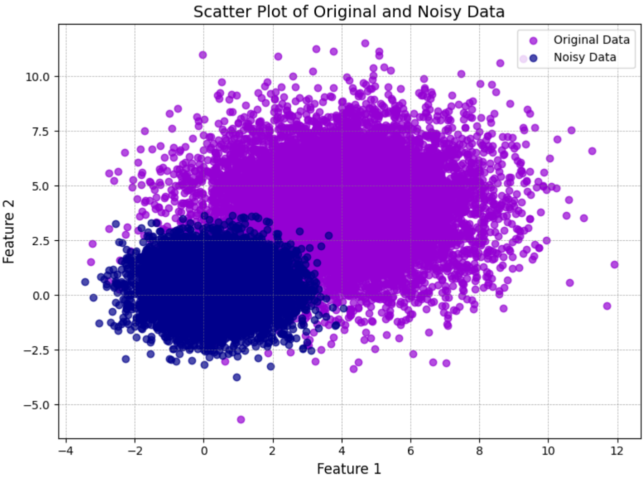
$$E_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2 \right]$$

where $\tilde{\mu}$ is the optimal mean that depends on x_0 which we don't know. We can also have ϵ using

$$E_q \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\| \right]$$

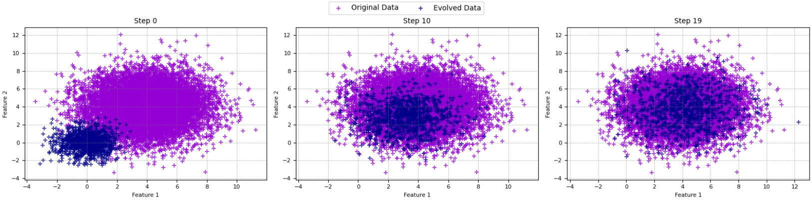
Our results - Gaussian

We have started with Gaussian generation:



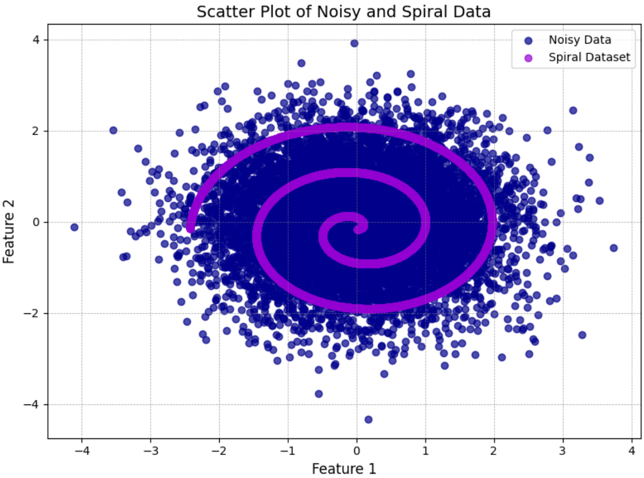
Our results - Gaussian

We have started with Gaussian generation and got satisfying results:



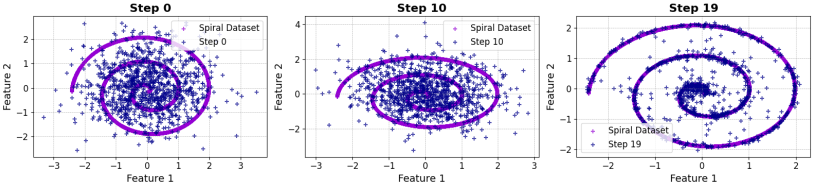
Our results - Spirale

Then we moved to a more complicated dataset, Spirale generation:



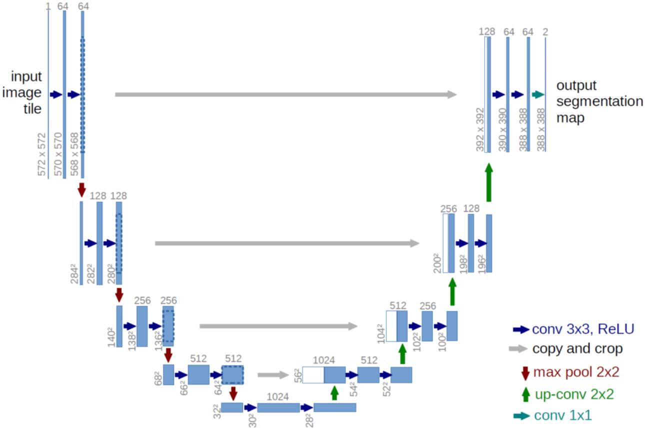
Our results - Spirale

Then we moved to a more complicated dataset, Spirale generation and also got satisfying results:

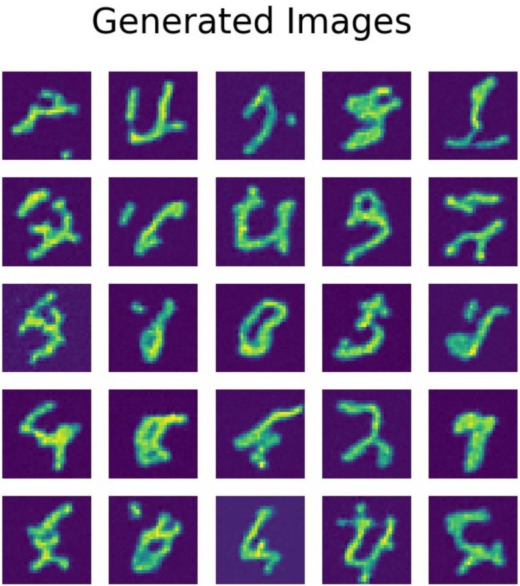


Generating Images

We need a model powerful enough to predict ϵ (or μ) for 2D images (the size of ϵ), for this we use UNet.



Our results - MNIST Generation



Amelioration

Accelerating generation

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Imagine that you have $T = 10,000$. You can not do 10,000 steps to generate one image. But, using the reparametrization trick:

$$x_T = x_0 + \sigma_{T,0}\epsilon_T \Rightarrow \tilde{x_0}$$

$$x_{T-k} = \tilde{x_0} + \sigma_{T-k,0}\epsilon_{T-k}$$

So we can generate with T/k steps.

OpenAI's incrementation

Improved Denoising Diffusion Probabilistic Models

Alex Nichol [*,1] **Prafulla Dhariwal** [*,1]

OpenAI’s incrementation

Improved Denoising Diffusion Probabilistic Models

Alex Nichol [*1] **Prafulla Dhariwal** [*1]

Recall that:

- Σ_θ was set to a constant,

OpenAI's incrementation

Improved Denoising Diffusion Probabilistic Models

Alex Nichol [*1] **Prafulla Dhariwal** [*1]

Recall that:

- Σ_θ was set to a constant,
- The training loss was a simplified version of reality.

OpenAI's incrementation

Improved Denoising Diffusion Probabilistic Models

Alex Nichol [*1] **Prafulla Dhariwal** [*1]

Recall that:

- Σ_θ was set to a constant,
- The training loss was a simplified version of reality.

This paper tackle these problems.

How to modelize Σ_θ

Ho et al. fixed $\Sigma_\theta(x_t, t) = \sigma_t^2 I$ with:

- $\sigma_t^2 = \beta_t$ optimal if $x_0 \sim \mathcal{N}(0, I)$,

How to modelize Σ_θ

Ho et al. fixed $\Sigma_\theta(x_t, t) = \sigma_t^2 I$ with:

- $\sigma_t^2 = \beta_t$ optimal if $x_0 \sim \mathcal{N}(0, I)$,
- $\sigma_t^2 = \bar{\beta}_t$ optimal if x_0 is a point.

How to modelize Σ_θ

Ho et al. fixed $\Sigma_\theta(x_t, t) = \sigma_t^2 I$ with:

- $\sigma_t^2 = \beta_t$ optimal if $x_0 \sim \mathcal{N}(0, I)$,
- $\sigma_t^2 = \bar{\beta}_t$ optimal if x_0 is a point.

Hence for each t, we know that $\Sigma^*(x_, t)$ is between β_t and $\bar{\beta}_t$.

How to modelize Σ_θ

Ho et al. fixed $\Sigma_\theta(x_t, t) = \sigma_t^2 I$ with:

- $\sigma_t^2 = \beta_t$ optimal if $x_0 \sim \mathcal{N}(0, I)$,
- $\sigma_t^2 = \bar{\beta}_t$ optimal if x_0 is a point.

Hence for each t, we know that $\Sigma^*(x_, t)$ is between β_t and $\bar{\beta}_t$.

Ho et al. have found that the impact is negligeable...

How to modelize Σ_θ

Hence for each t, we know that $\Sigma^*(x,t)$ is between β_t and $\bar{\beta}_t$.

Ho et al. have found that the impact is negligeable...

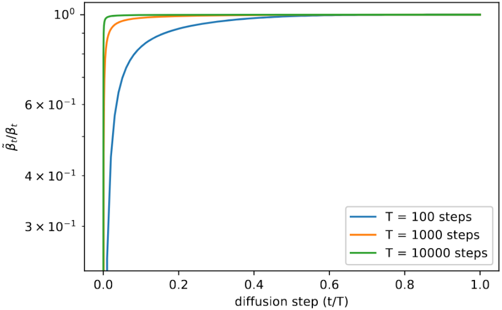


Figure 1. The ratio $\tilde{\beta}_t/\beta_t$ for every diffusion step for diffusion processes of different lengths.

How to modelize Σ_θ

Hence for each t, we know that $\Sigma^*(x_, t)$ is between β_t and $\bar{\beta}_t$.

Ho et al. have found that the impact is negligeable. But it depends of other hyperparameters.

Hence, they interpolate between the two extreme values, and let the model learn $v(t)$:

$$\Sigma_\theta(x_t, t) := \exp(v_\theta(t) \log(\beta_t) + (1 - v_\theta(t)) \log(\bar{\beta}_t))$$

Changing the Loss

In the paper of Ho, they use the loss L_{simple}, a simplified version of L_{LVB}, that only affects μ_θ.

Changing the Loss

In the paper of Ho, they use the loss L_{simple}, a simplified version of L_{LVB}, that only affects μ_θ.

Optimizing L_{simple} performs well for FID, but poorly for log-likelihood metric.

Changing the Loss

In the paper of Ho, they use the loss L_{simple}, a simplified version of L_{LVB}, that only affects μ_θ.

Optimizing L_{simple} performs well for FID, but poorly for log-likelihood metric.

L_{LVB} affects Σ_θ but increasing log-likelihood score can increase FID, hence we need a trade-of.

Changing the Loss

In the paper of Ho, they use the loss L_{simple}, a simplified version of L_{LVB}, that only affects μ_θ.

Optimizing L_{simple} performs well for FID, but poorly for log-likelihood metric.

L_{LVB} affects Σ_θ but increasing log-likelihood score can increase FID, hence we need a trade-of.

They train on

$$L = L_{simple} + \lambda L_{LVB}$$

Changing the Loss

In the paper of Ho, they use the loss L_{simple}, a simplified version of L_{LVB}, that only affects μ_θ.

Optimizing L_{simple} performs well for FID, but poorly for log-likelihood metric.

L_{LVB} affects Σ_θ but increasing log-likelihood score can increase FID, hence we need a trade-of.

They train on

$$L = L_{simple} + \lambda L_{LVB}$$

This loss is prone to gradient exploding and we need importance sampling to implement it.

Classifier Guidance

Importance of labels

Let's get back to hand-written digits generation:

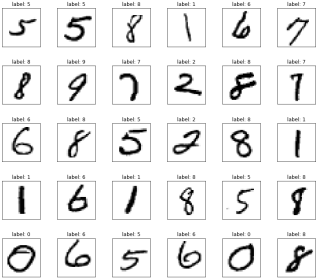


Figure 1: Source: ludwig.ai

Importance of labels

Let's get back to hand-written digits generation:

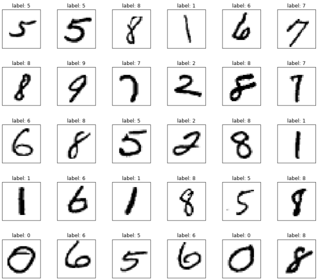


Figure 1: Source: ludwig.ai

A DDPM can generate new images that look like digits, but the model can't distinguish a mix of two digits and a real digit.

Importance of labels

Let's get back to hand-written digits generation:

If we have a classifer that gives $p_\phi(y \mid x_t)$, we can sample using

$$p_{\theta,\phi}(x_t \mid x_{t+1}, y) = Z p_\theta(x_t \mid x_{t+1}) p_\phi(y \mid x_t)$$

This way if we set $y = 3$, we can trick our model to generate somthing that looks like a 3.

Langevin Dynamics

Given $x_0 \sim \pi(x)$ an unknown distribution, if we iterate through $x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i$ with $\epsilon \to 0$ and $z_i \sim \mathcal{N}(0, I)$, we can sample from $p(x)$.

Langevin Dynamics

Given $x_0 \sim \pi(x)$ an unknown distribution, if we iterate through $x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i$ with $\epsilon \to 0$ and $z_i \sim \mathcal{N}(0, I)$, we can sample from $p(x)$.

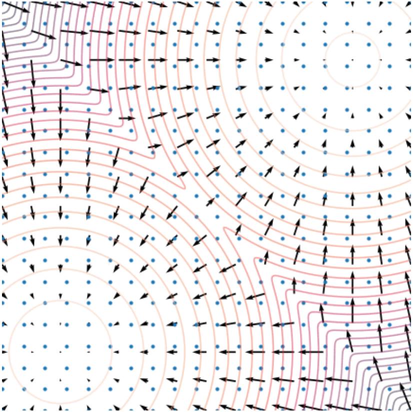


Figure 1: Visualizations from Yang Song's work.

Langevin Dynamics

Given $x_0 \sim \pi(x)$ an unknown distribution, if we iterate through $x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i$ with $\epsilon \to 0$ and $z_i \sim \mathcal{N}(0, I)$, we can sample from $p(x)$.

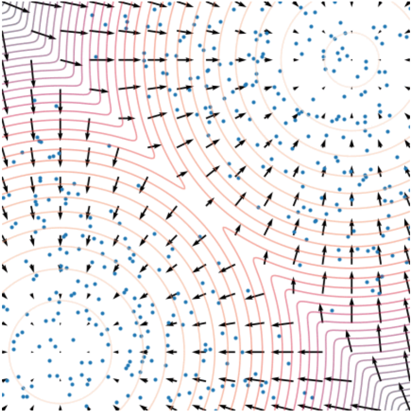


Figure 1: Visualizations from Yang Song's work.

Langevin Dynamics

Given $x_0 \sim \pi(x)$ an unknown distribution, if we iterate through $x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i$ with $\epsilon \to 0$ and $z_i \sim \mathcal{N}(0, I)$, we can sample from $p(x)$.

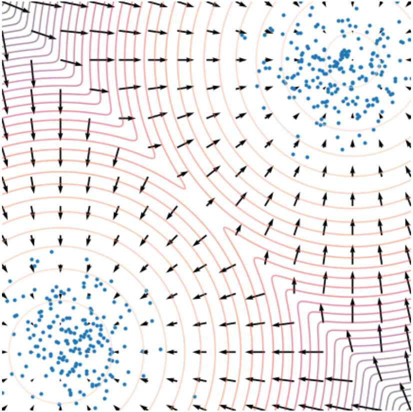


Figure 1: Visualizations from Yang Song's work.

Langevin Dynamics

Given $x_0 \sim \pi(x)$ an unknown distribution, if we iterate through $x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i$ with $\epsilon \to 0$ and $z_i \sim \mathcal{N}(0, I)$, we can sample from $p(x)$.

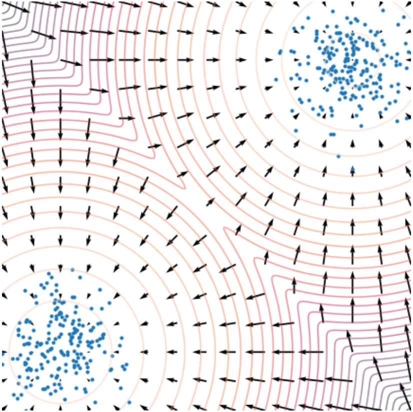


Figure 1: Visualizations from Yang Song's work.

Langevin Dynamics

Given $x_0 \sim \pi(x)$ an unknown distribution, if we iterate through $x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i$ with $\epsilon \to 0$ and $z_i \sim \mathcal{N}(0, I)$, we can sample from $p(x)$.

So we need to know $\nabla_x \log p(x)$, but don't need to know $p(x)$.

Langevin Dynamics

Given $x_0 \sim \pi(x)$ an unknown distribution, if we iterate through $x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i$ with $\epsilon \to 0$ and $z_i \sim \mathcal{N}(0, I)$, we can sample from $p(x)$.

So we need to know $\nabla_x \log p(x)$, but don't need to know $p(x)$.

From the classifier p_ϕ, one can get an approximation of $\nabla_x \log p(x)$.

Low vs High temperature

Figure 1: Classifier-free Diffusion Guidance

Low vs High temperature

Figure 1: Classifier-free Diffusion Guidance

- Low-temperature optimizes FID score

Low vs High temperature

Figure 1: Classifier-free Diffusion Guidance

- Low-temperature optimizes FID score
- High-temperature optimizes Inception score

Low vs High temperature

Figure 1: Classifier-free Diffusion Guidance

- Low-temperature optimizes FID score
- High-temperature optimizes Inception score

We can do a trade-of by following more p_ϕ or p_θ between exploration and distance to the original distribution.

GLIDE: draw what you prompt

Classifier-Free Guidance

Previous guidance need a trained classifier. They define:

- An unconditional DDPM, that predicts $p_\theta(z)$.
- A conditional DPPM, that predicts $p_\theta(z \mid c)$.

Classifier-Free Guidance

Previous guidance need a trained classifier. They define:

- An unconditional DDPM, that predicts $p_\theta(z)$.
- A conditional DPPM, that predicts $p_\theta(z \mid c)$.

Rather than using p_ϕ, they train both models simultanously and use the gradient of $p_\theta(z \mid c)$ to estimate $\nabla \log p(z \mid c)$.

CLIP

To force a label c, we train a model that estimates the probability for x to be of class c, and uses its gradient.

CLIP

To force a label c, we train a model that estimates the probability for x to be of class c, and uses its gradient.

To force a sentence c, we train a model that estimates the distance between x and the sentence c, and uses its gradient.

CLIP

To force a sentence c, we train a model that estimates the distance between x and the sentence c, and uses its gradient.

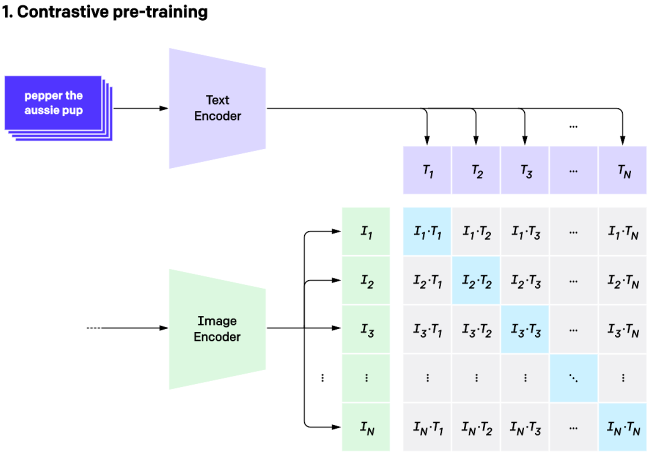
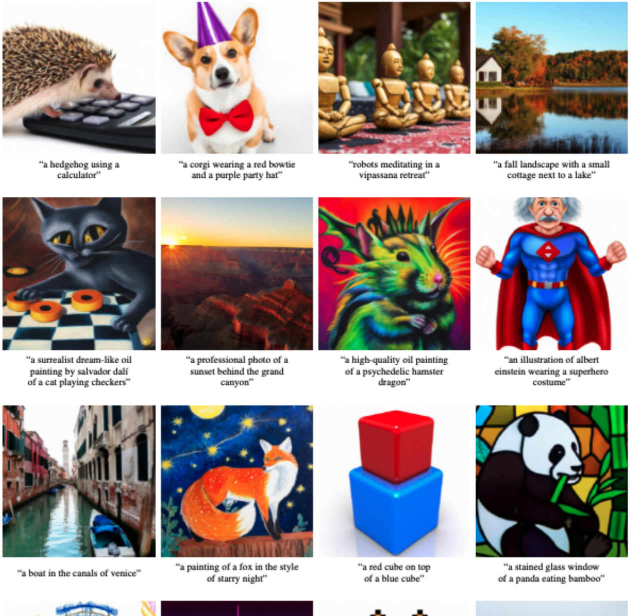


Figure 2: How to compare an image and a text

Results (OpenAI GLIDE)



"a hedgehog using a calculator"

"a corgi wearing a red bowtie and a purple party hat"

"robots meditating in a vipassana retreat"

"a fall landscape with a small cottage next to a lake"

"a surrealist dream-like oil painting by salvador dalí of a cat playing checkers"

"a professional photo of a sunset behind the grand canyon"

"a high-quality oil painting of a psychedelic hamster dragon"

"an illustration of albert einstein wearing a superhero costume"

"a boat in the canals of venice"

"a painting of a fox in the style of starry night"

"a red cube on top of a blue cube"

"a stained glass window of a panda eating bamboo"

State-of-the-art brief review

ControlNet

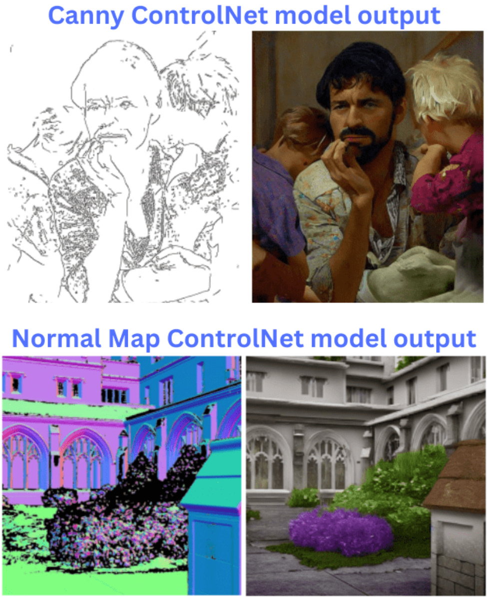
GLIDE accepts one more input than text: a mask for inpainting.

ControlNet

GLIDE accepts one more input than text: a mask for inpainting.

ControlNet generalises it by adding more optional inputs (e.g. Cany edges representation / Human Pose / Sketch).

ControlNet


Canny ControlNet model output
Normal Map ControlNet model output