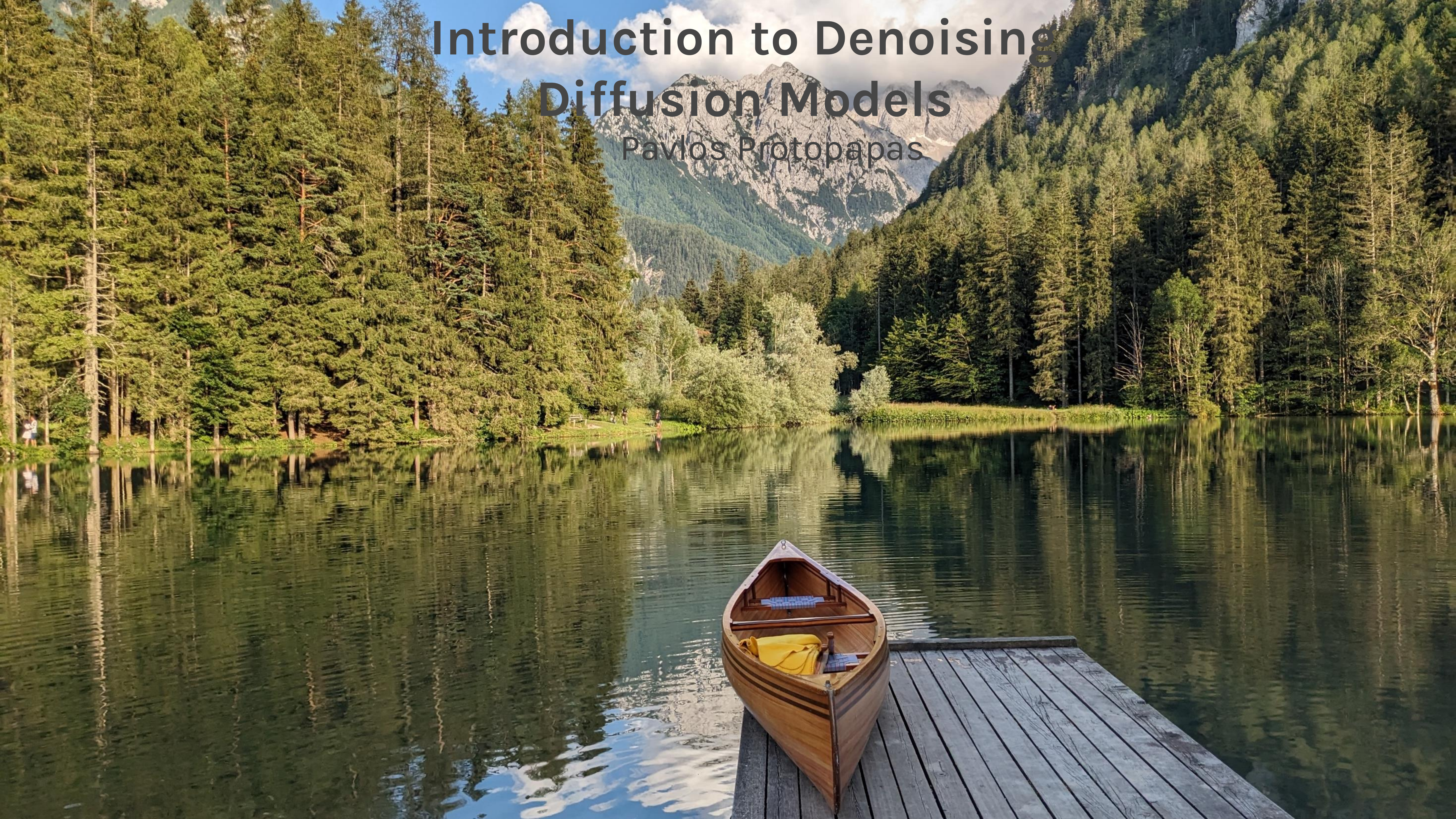
Introduction to Denoising Diffusion Models

Pavlos Protopapas



Outline

- Motivation for Diffusion Models
- Introduction to Denoising Diffusion Models
- Training a diffusion model
- Stable Diffusion
- DALL-E 2

Outline

- **Motivation for Diffusion Models**
- Introduction to Denoising Diffusion Models
- Training a diffusion model
- Stable Diffusion
- DALL-E 2

Motivation for Diffusion Models

Motivation for Diffusion Models

Denoising diffusion models have shown impressive sample quality and diversity for several image generation tasks.

Stable Diffusion V2

Motivation for Diffusion Models

Denoising diffusion models have shown impressive results for text-to-image tasks.

[DALL-E 2]

"A Univ.AI student about to write a paper on neural differential equations painted in a pointilistic style"

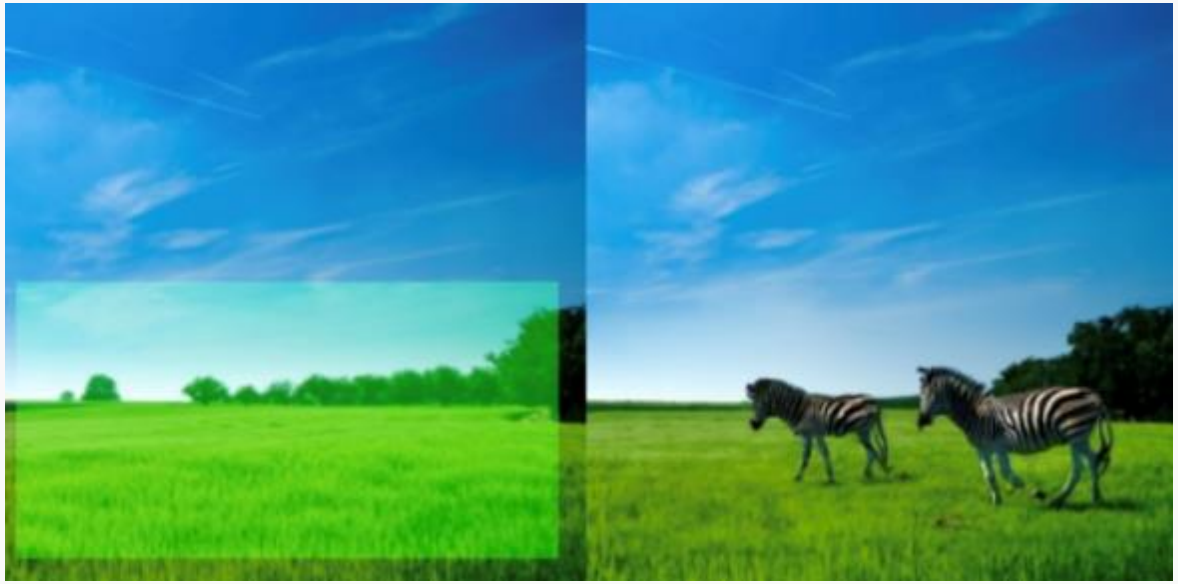
[Imagen]

"A robot couple fine dining with the Eiffel Tower in the background"

Motivation for Diffusion Models

Denoising diffusion models have shown impressive results for image editing and inpainting tasks.

“A man with red hair”

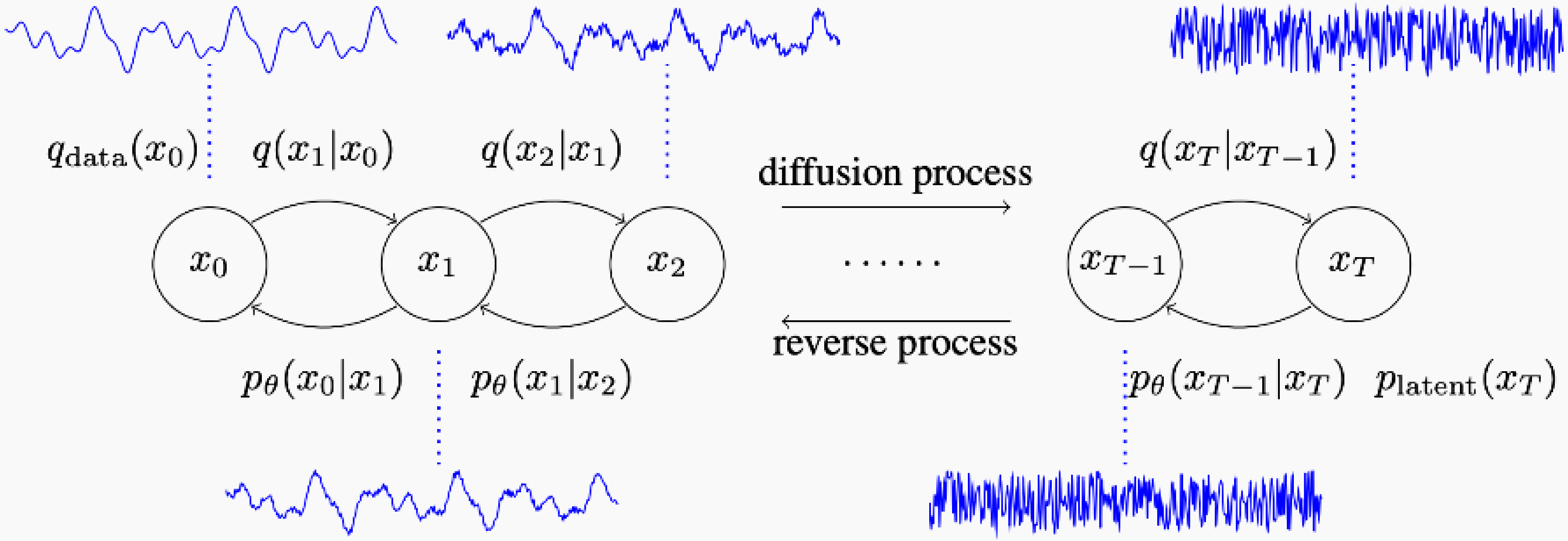


“Zebras roaming the field”

GLIDE

Motivation for Diffusion Models

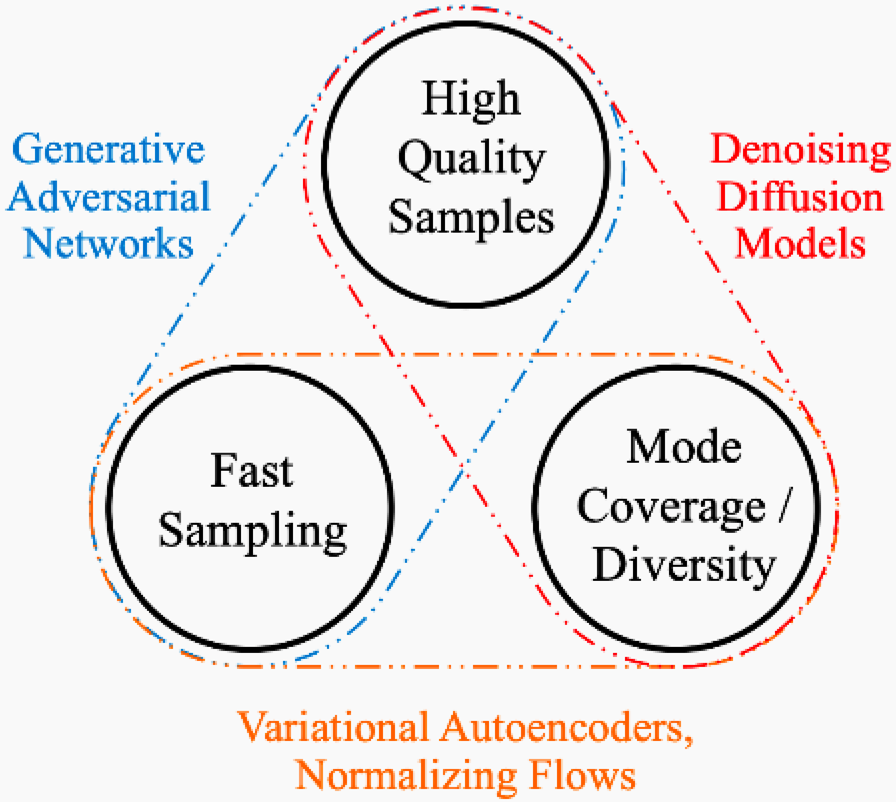
Denoising diffusion models have also been effective in non-visual domains.



[Diffwave: A versatile diffusion model for audio synthesis]

Motivation for Diffusion Models

But are denoising diffusion models all you need?



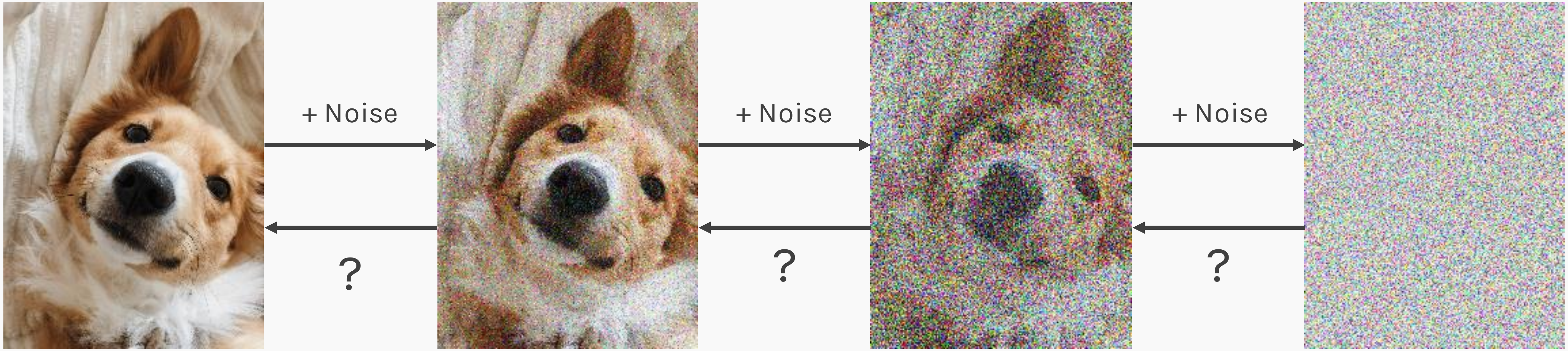
Generative learning trilemma
[Source]

Outline

- Motivation for Diffusion Models
- **Introduction to Denoising Diffusion Models**
- Training a diffusion model
- Stable Diffusion
- DALL-E 2

Denoising Diffusion Models

The general idea of diffusion models is quite simple.

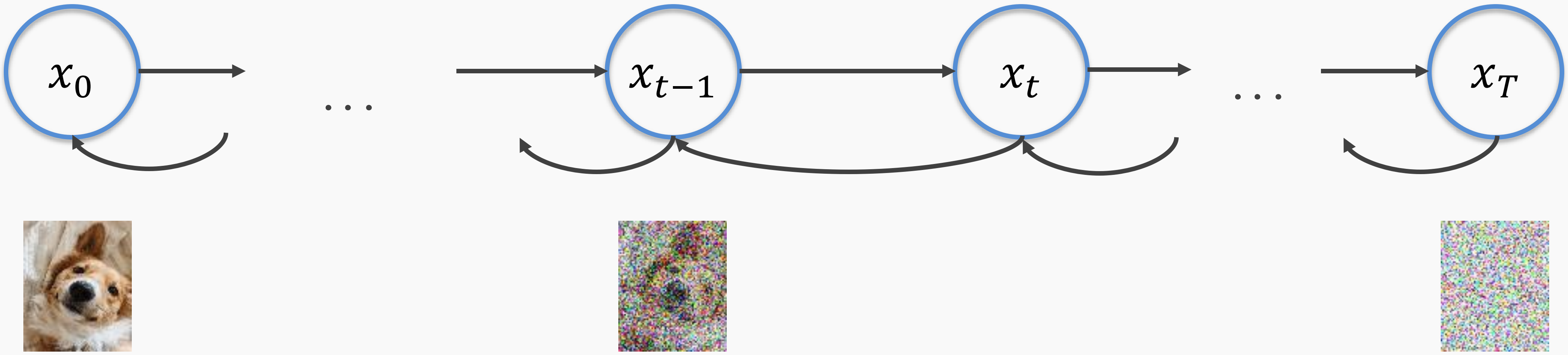


Diffusion models help undoing this process: Starting with noise, it can gradually move toward a coherent image.

Denoising Diffusion Models

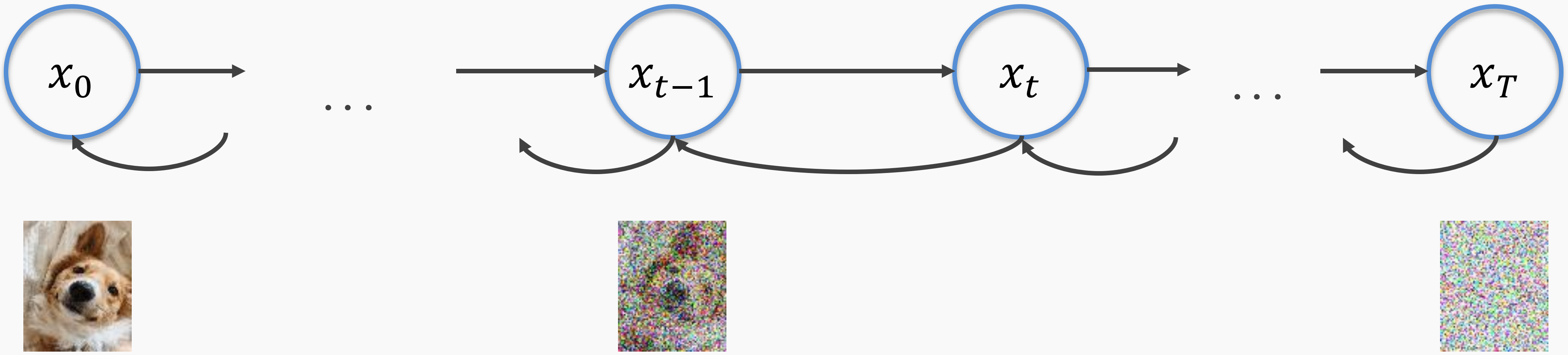
A Markov chain means that each step only depends on the previous one.

In practice, they are formulated using a Markov chain of T steps.



- They take the input image and gradually add Gaussian noise to it through a series of small steps.
- A neural network is trained to recover the original data by reversing the noising process. By being able to model the reverse process, we can generate new data.

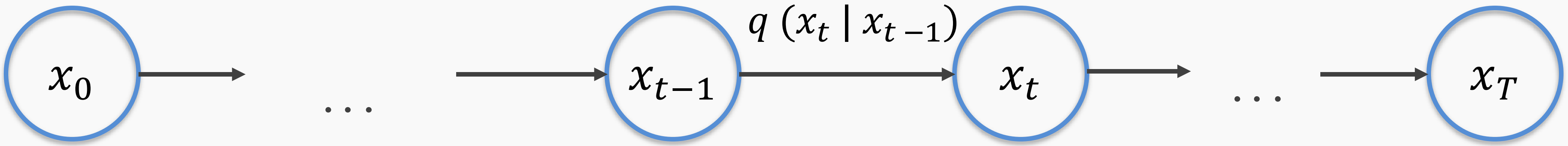
Denoising Diffusion Models



Therefore, there are two processes in a diffusion model:

1. Forward process
2. Reverse process

Denoising Diffusion Models – Forward Process



Given a data point x_0 sampled from the real data distribution $q(x)$.

At each forward step of the Markov chain, we add a small amount of Gaussian noise with variance β_t to x_{t-1}, producing a new latent variable x_t with distribution $q\ (x_t \mid x_{t-1})$.
This diffusion process can be formulated as follows:

$$q\ (x_t \mid x_{t-1}) = N(\mu_t = \sqrt{1-\beta_t}x_{t-1}, \Sigma_t = \beta_t\ I\ )$$

As T→ ∞, you eventually end up with an isotropic gaussian (i.e. pure random noise ~ $N(0, I)$).

Denoising Diffusion Models – Forward Process

In general:

$$q\ (x_{1:T} \mid x_0) = \prod_{t=1}^{T} q\ (x_t \mid x_{t-1})$$

where:

$$q\ (x_t \mid x_{t\ -1}) = N(\mu_t = \sqrt{1\ - \beta_t} x_{t-1},\ \Sigma_t = \beta_t\ I\ )$$

But how would you find $q\ (x_{1000} \mid x_0)$?

You would have to go from x_0 till x_{999} step-by-step until you can compute $q\ (x_{1000} \mid x_{999})$!

Denoising Diffusion Models – Forward Process

By reparametrizing such that:

$$\alpha_t = 1 - \beta_t$$

$$\overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$$

We get:

$$q\,(x_t \mid x_0) = N(\mu_t = \sqrt{\overline{\alpha_t}}\,x_0,\ \Sigma_t = (1 - \overline{\alpha_t})\,I\,)$$

Using the above equation, we can now generate any time step t directly from x_0 instead of iteratively going through x_{t-1} , x_{t-2} … and so on.

Denoising Diffusion Models – Forward Process

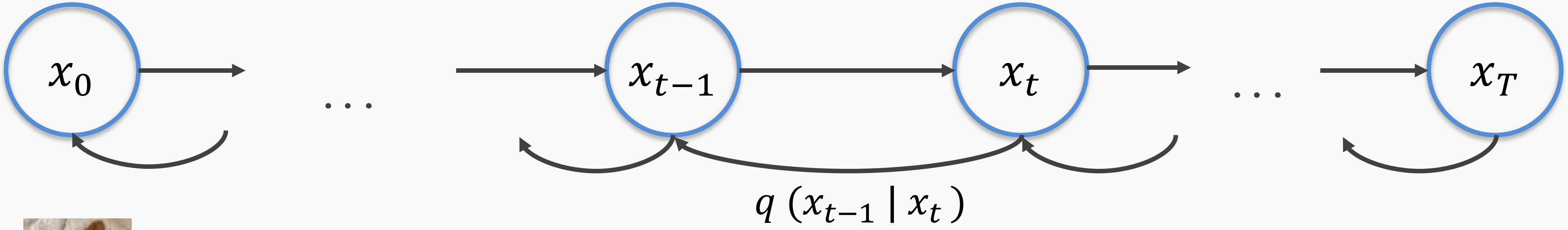
- The variance parameter β_t can be fixed to a constant or chosen as a schedule over the T timesteps.

- The original DDPM authors utilized a linear schedule, however the Improving DDPM paper showed that employing a cosine schedule works even better.

Linear Scheduler (top), Cosine scheduler (bottom)

Source

Denoising Diffusion Models – Reverse Process



- The goal of a diffusion model is to learn a neural network to reverse the denoising process by iteratively undoing the forward process.

- If we learn the reverse distribution $q\,(x_{t-1}\,|\,x_t\,)$, we can then:
  1. Sample x_T from $N(0, I)$
  2. Run the reverse process iteratively and acquire a sample from $q\,(x_0)$, generating a novel data point from the original data distribution.

Denoising Diffusion Models – Reverse Process

But finding $q\ (x_{t-1} \mid x_t\ )$ is difficult as it depends on the entire data distribution and is therefore intractable.

Turns out that for small enough β_t, $q\ (x_{t-1} \mid x_t\ )$ will also be Gaussian.

We can then approximate $q\ (x_{t-1} \mid x_t\ )$ with a gaussian distribution $p_\theta(x_{t-1} \mid x_t)$ which is parameterized by some θ:

$$p_\theta(x_{t-1} \mid x_t) = N(\mu_\theta(x_t, t),\ \textstyle\sum_\theta (x_t, t))$$

and

$$p_\theta(x_{0:T})\ = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$$

Denoising Diffusion Models – Reverse Process

How do you train a model to learn these parameters θ? What is the loss function we need to optimize?

We can use the maximum likelihood estimate by optimizing the negative log-likelihood of the training data.

Using variational inference and the evidence lower bound, we get our loss to be:

$$L = \mathbb{E}_{q(x_0:T)}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

On further simplification, we get:

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

Denoising Diffusion Models – Reverse Process

$$L = \mathbb{E}[\underbrace{D_{KL}(q(x_T|x_0) \parallel p(x_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta (x_{t-1}|x_t))}_{L_{t-1}} \underbrace{- \log \mathrm{p}_\theta(x_0|x_1)}_{L_0}]$$

Therefore,

$$L = L_T + L_{T-1} + L_{T-2} + \ \ldots + L_0$$

where

$$L_T = D_{KL}\big(q(x_T|x_0) \parallel p(x_T)\big)$$

$$L_t = D_{KL}(q(x_t|x_{t+1}, x_0) \parallel p_\theta (x_t|x_{t+1}) \text{ for } 1 \leq \mathrm{t} \leq \mathrm{T} - 1$$

$$\mathrm{L}_0 = -\log \mathrm{p}_\theta(x_0|x_1)$$

Denoising Diffusion Models – Reverse Process

$$L = \mathbb{E}[\underbrace{D_{KL}(q(x_T|x_0) \parallel p(x_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))}_{L_{t-1}} \underbrace{- \log \mathrm{p}_\theta(x_0|x_1)}_{L_0}]$$

Here, L_T is constant and does not vary with the parameter we are trying to optimize over θ.

Thus, we can ignore this term to get:

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

Denoising Diffusion Models – Reverse Process

Looking only at term $L_t = D_{KL}(q(x_t|x_{t+1}, x_0) \parallel p_\theta(x_t|x_{t+1})$ for $1 \leq \mathrm{t} \leq \mathrm{T} - 1$

$$q(x_t|x_{t+1}, x_0) = N(\widetilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$$

Where we reparametrize the normal distribution:

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \overline{\alpha_t}}}\epsilon\right)$$

$$\tilde{\beta}_t = \beta_t\left(\frac{1 - \overline{\alpha_{t-1}}}{1 - \overline{\alpha_t}}\right)$$

where

$$\alpha_t = 1 - \beta_t, \qquad \overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$$

$\epsilon_t \sim \mathrm{N}(0, \mathrm{I})$, β_t is the variance scheduler

$$p_\theta(x_{t-1} \mid x_t) = N(\mu_\theta(x_t, t), \textstyle\sum_\theta(x_t, t))$$

where

$\mu_\theta(x_t, t)$ is learnt by the neural network

$\sum_\theta(x_t, t)$ is fixed as $\tilde{\beta}_t$ or β_t

Denoising Diffusion Models – Reverse Process

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon\right)$$

$\mu_\theta(x_t, t)$ is learnt by the neural network

Our goal is to now reduced to ensuring that $\mu_\theta(x_t, t) \approx \tilde{\mu}_t$.

We can do this by minimizing the MSE between the two:

$$L_t = \frac{1}{2\sigma^2} \parallel \tilde{\mu}_t - \mu_\theta(x_t, t) \parallel^2$$

$$= \frac{1}{2\sigma^2} \parallel \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon\right) - \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon_\theta(x_t, t)\right) \parallel^2$$

Denoising Diffusion Models – Reverse Process

$$= \frac{1}{2\sigma^2} \| \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon\right) - \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon_\theta(x_t, t)\right) \|^2$$

$$L_t = \frac{\beta_t^2}{2\sigma^2\alpha_t(1-\overline{\alpha_t})} \| \epsilon - \epsilon_\theta(x_t, t) \|^2$$

The authors have found that dropping the constant term have led to better sample quality, so:

$$L_t = \| \epsilon - \epsilon_\theta(x_t, t) \|^2$$

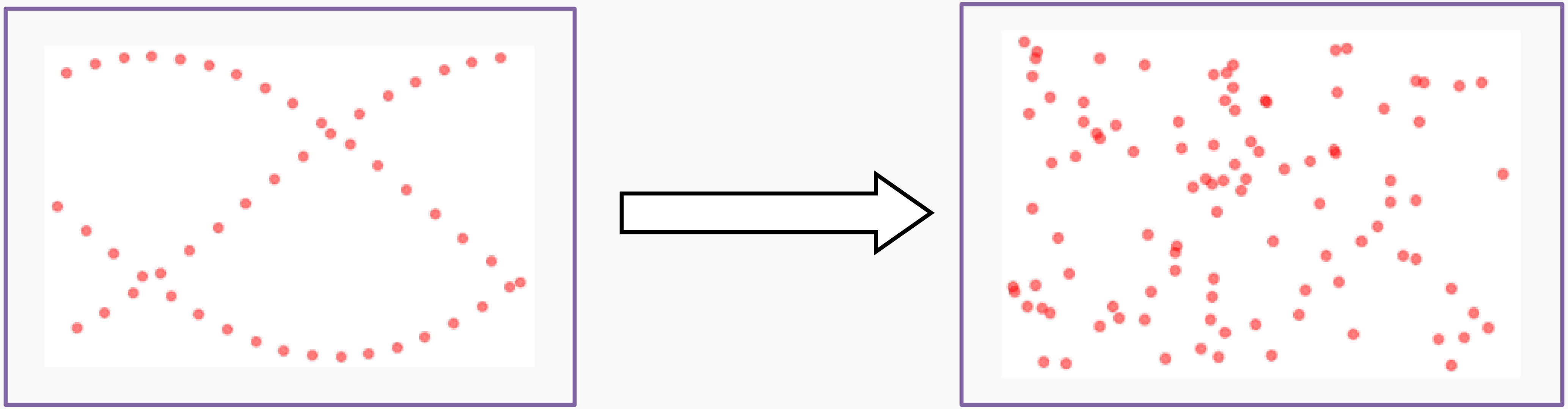
where $\mathrm{x}_t = \sqrt{\overline{\alpha_t}}x_0, + \sqrt{1-\overline{\alpha_t}}\epsilon$

Outline

- Motivation for Diffusion Models
- Introduction to Denoising Diffusion Models
- **Training a diffusion model**
- Stable Diffusion
- DALL-E 2

Training a diffusion model

Abstractly, our goal is to learn noise from structured data, which is nontrivial.



Structured data

Noise

Training a diffusion model

Now that we have found our loss function:

$$L_t = \| \epsilon - \epsilon_\theta(x_t, t) \|^2$$

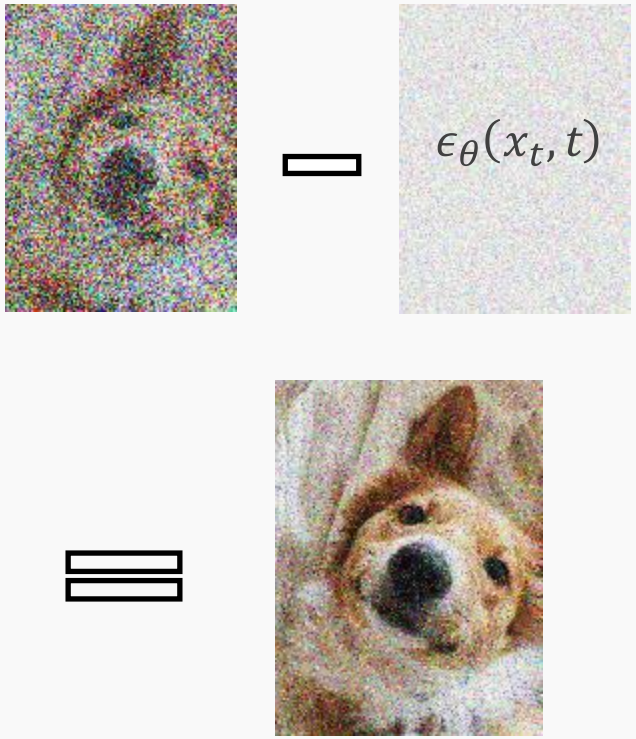
where $x_t = \sqrt{\overline{\alpha_t}}x_0, + \sqrt{1 - \overline{\alpha_t}}\,\epsilon$

We can use a neural network to take in x_t and predict a noise $\epsilon_\theta(x_t, t)$ that describes the noise that was added to x_{t-1} to get x_t.

Therefore, by passing x_t through our U-Net architecture, we get a noise that we can subtract from x_t to get x_{t-1}. By doing this iteratively, we can get back x_0.

Given x_t, the model must predict the noise $\epsilon_\theta(x_t, t)$ such that

$x_t - \epsilon_\theta(x_t, t)$ = x_{t-1}



Training a diffusion model

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

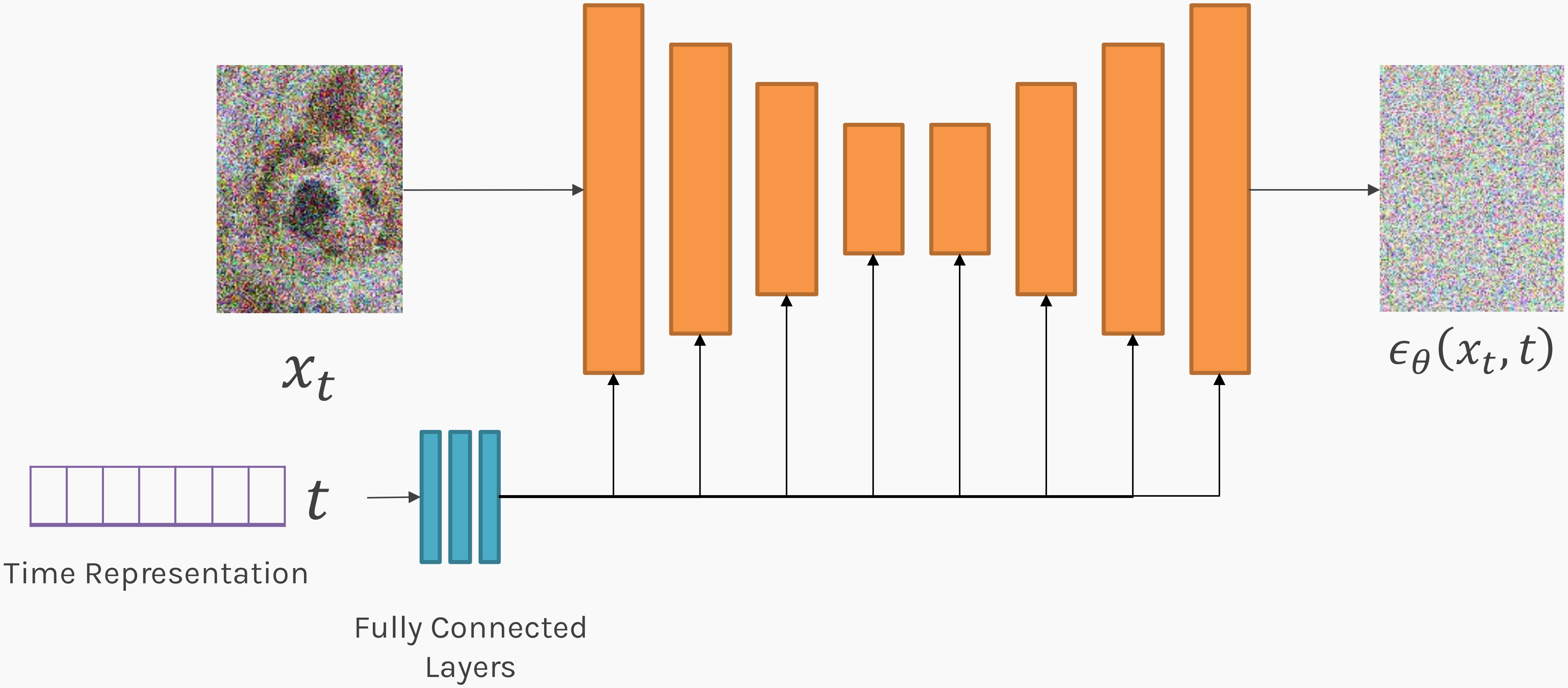
Training a diffusion model

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

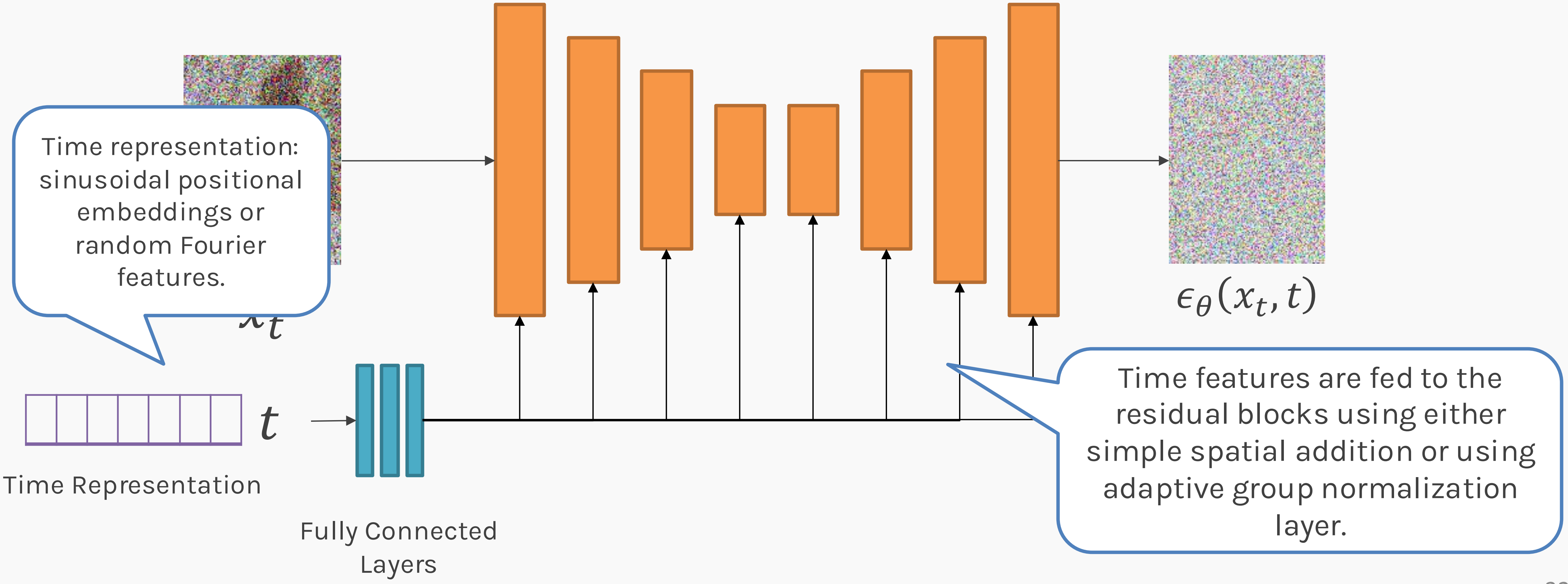
Training a diffusion model

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(x_t, t)$.



Training a diffusion model

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(x_t, t)$.



Outline

- Motivation for Diffusion Models
- Introduction to Denoising Diffusion Models
- Training a diffusion model
- **Stable Diffusion**
- DALL-E 2

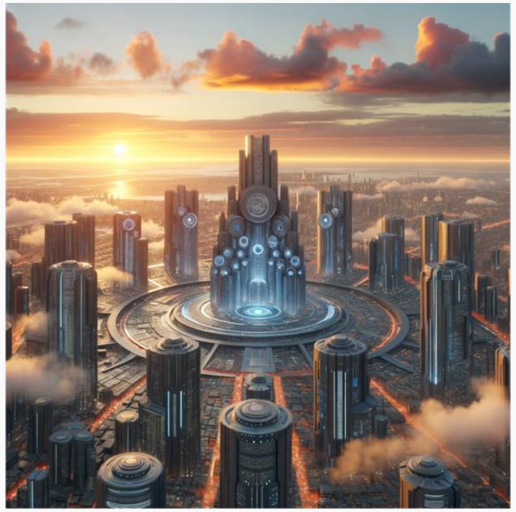
Examples of recent diffusion models

There are many recent text-to-image or image generation diffusion models:

- DALL-E 2
- DALL-E 3
- GLIDE
- Imagen
- Imagen 2
- MidJourney
- Stable Diffusion 1.x
- Stable Diffusion 2.x
- Adobe Firefly
- And many more!

"A futuristic cityscape at sunset."


Imagen 2


DALL-E 3


Adobe Firefly

In today's class, we will be talking about Stable Diffusion and DALL-E 2, once you understand this, you can understand other diffusion models!

Latent Diffusion Models

- Training models in pixel space is very computationally expensive as images are high dimensional data.

- Latent diffusion models tackle this issue. Instead of applying the diffusion process directly on a high-dimensional input, they project the input into a smaller latent space and apply the diffusion there.

Latent Diffusion Models

There are two stages of latent diffusion models:

1. Train perceptual compression models to learn a semantically equivalent latent space by removing high-level details. The loss includes reconstruction, adversarial, and regularization terms.

2. Perform diffusion process in the latent space for efficiency and focusing on relevant semantic data bits.

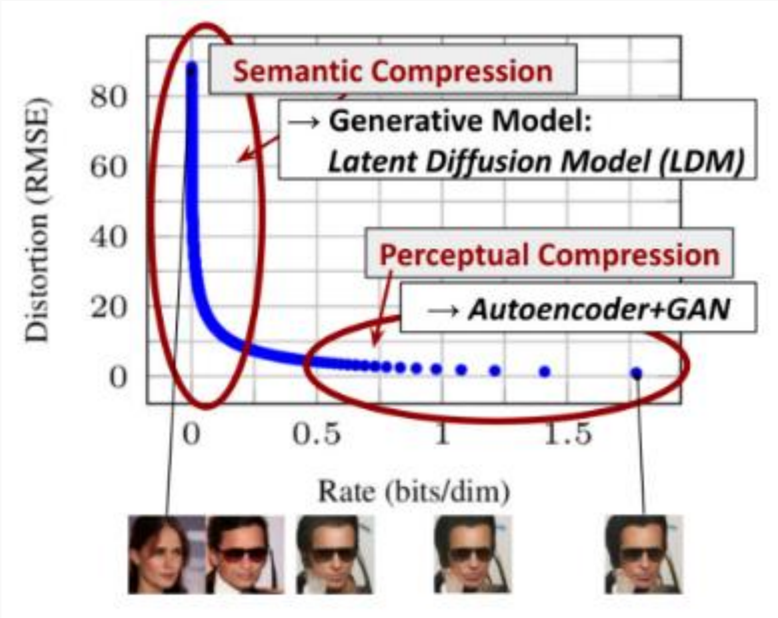


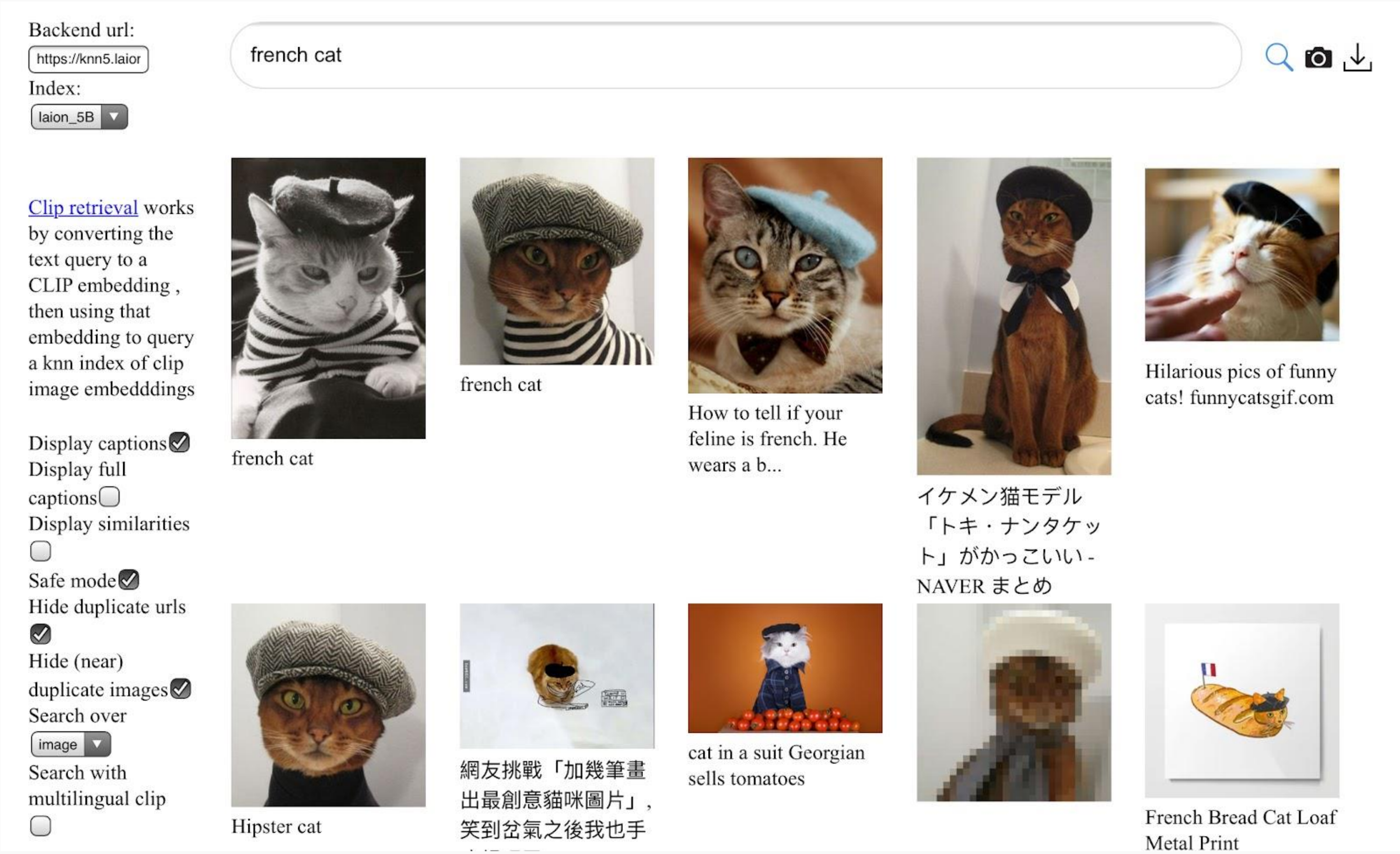
Image from Rombach et al. in *High-resolution image synthesis with latent diffusion models*

Stable Diffusion (R Rombach et al. 2021)

- Stable Diffusion (SD) is a family of open-source conditional **latent diffusion** models, able to generate high-quality images from various conditions (such as textual descriptions).

Stable Diffusion

- The model was trained to generate realistic images from textual descriptions, using a subset of LAION-5B, which is an open dataset of (image, caption) pairs.



Stable Diffusion – The different parts

A **Variational Autoencoder** (VAE) that is used to both map an image to a lower dimensional latent representation (using the Encoder), and to go from the latent representation to an image (using the Decoder).

A **UNet** which is used to predict the noise added to an input image in each time step.

The **CLIP** model, particularly the text encoder, to incorporate textual information which are rich in image as well as conditional information for the model.

SD – Variational Autoencoder

- The main ideas behind latent diffusion models is to perform the diffusion process in a latent space, instead of directly using pixel space.

- In SD, an image $x_0 \in \mathbb{R}^{512 \times 512 \times 3}$ is first mapped using the **Encoder** to a low dimensional representation $z_0 \in \mathbb{R}^{64 \times 64 \times 4}$, where the diffusion process happens, obtaining a noisy latent z_T.

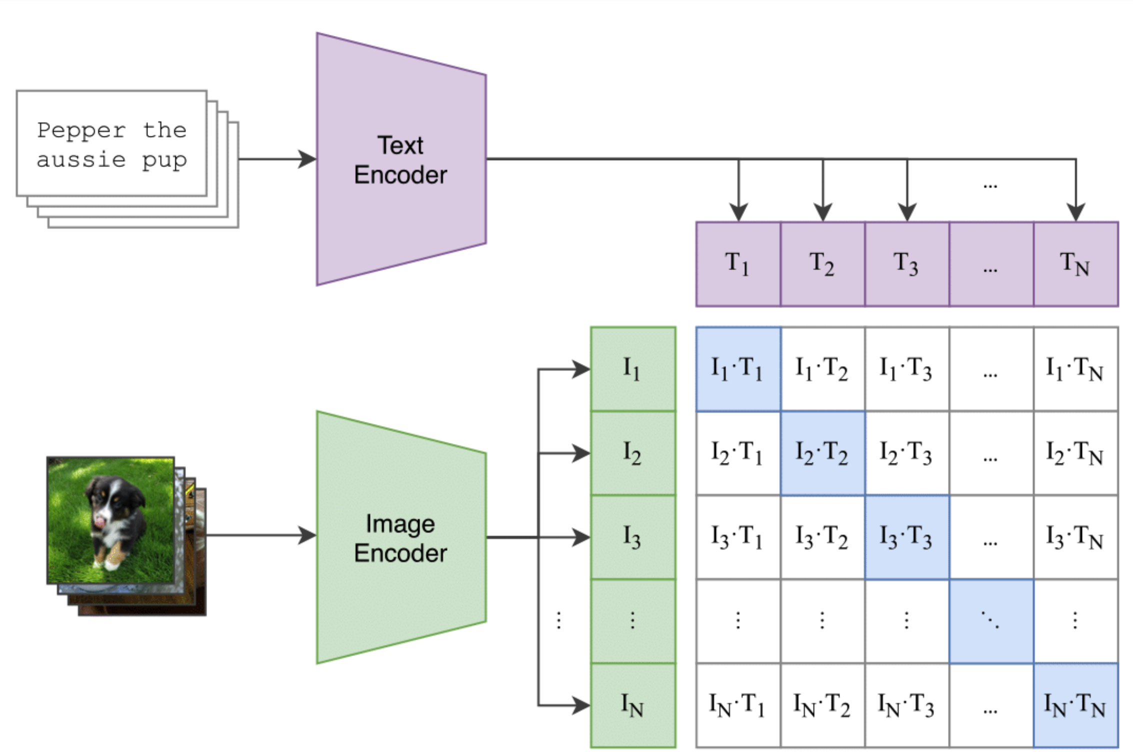
- Now the goal is to denoise z_T and map it back to an image using the **Decoder**.

- $512 \times 512 \times 3 \approx 768\text{k}$ and $64 \times 64 \times 4 \approx 16\text{k}$, this is a reduction of **48 times** in the dimension of the input!

SD – CLIP

- To be able to generate images conditioned on text, SD uses CLIP, a model jointly trained to learn image and text representations at the same time.

- The idea is that the text embeddings are rich in visual content as well and can help guide the image generation towards the given textual description.



SD – CLIP

- CLIP was trained on a dataset of 400 million (image, text) pairs, and is composed of an image transformer and a text transformer.

- During training if we have image text pairs $\{(x_i, y_i)\}_{i=1}^N$, we first use the transformers to obtain representations $\{(I_i, T_i)\}_{i=1}^N$.

- We then maximize the similarity between the embeddings of (image, text) pairs that are present in the dataset, and minimize the similarities of those which are not

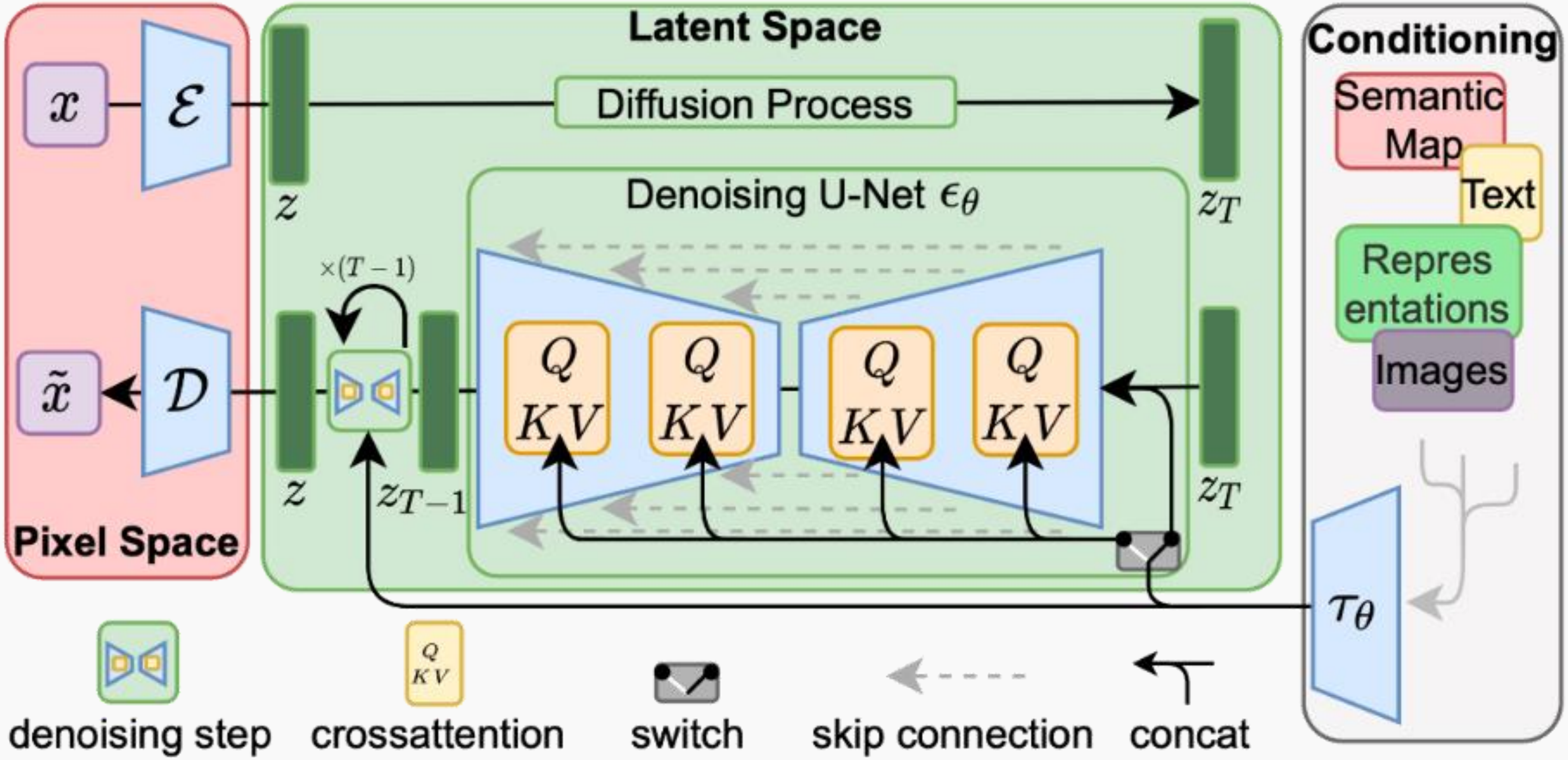
SD – UNet

- Finally, we have the UNet backbone, which it is the main part of the diffusion process.

- Given a a noisy latent z_t, a time step t and and the CLIP text embedding $\tau_\theta(y)$ of a prompt y, the UNet predicts the noise $\epsilon_\theta(z_t, t, \tau_\theta(y))$ which was added to go from z_{t-1} to z_t.

- The UNet uses cross-attention, where the queries refer to intermediate representations of the UNet, and the key and values to the conditional information $\tau_\theta(y)$

Stable Diffusion – Putting everything together



Outline

- Motivation for Diffusion Models
- Introduction to Denoising Diffusion Models
- Training a diffusion model
- Latent Diffusion Models
- **DALL-E 2**

DALL-E 2 (Ramesh et al. 2022)

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

DALL-E 2

From a high-level, DALL-E 2's works in three steps:

1. A text prompt is input into a **text encoder** that is trained to map the prompt y to a text encoding z_t.

2. A diffusion model $\mathrm{p}(z_i \mid y)$ called the **prior** maps the text encoding to a corresponding **image encoding** z_i that captures the semantic information of the prompt contained in the text encoding.

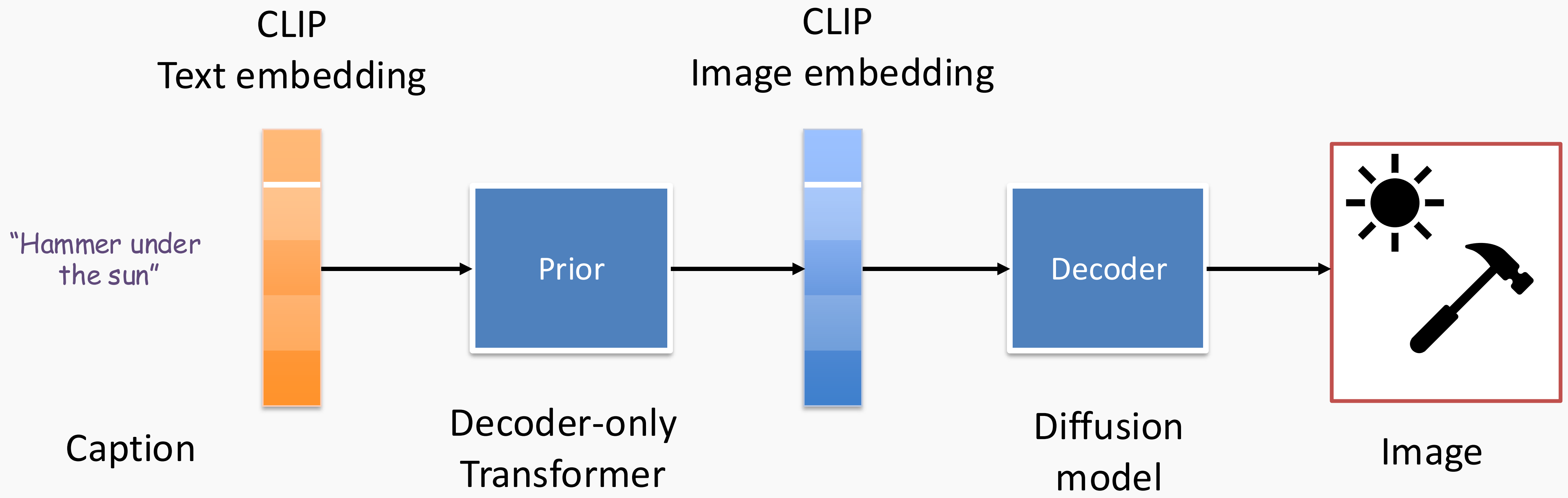
3. An **image decoder** $\mathrm{p}(\mathrm{x} \mid z_i, y)$ stochastically generates an image which is a visual manifestation of this semantic information.

4. The full conditional generative model is defined by:

$$\mathrm{p}(\mathrm{x} \mid \mathrm{y}) = \mathrm{p}(\mathrm{x} \mid z_i, y)\mathrm{p}(z_i \mid y)$$

DALL-E 2

- CLIP is used to map a caption y to a text embedding z_t which is fed to the prior to generate the image embedding z_i.
- The authors used both an autoregressive prior and a diffusion prior (decoder only transformer), but found the diffusion prior worked better.



DALL-E 2 Prior

- The use of a prior is not strictly necessary. For example, the model could condition only on the caption. However, the authors found that the prior increased image diversity.

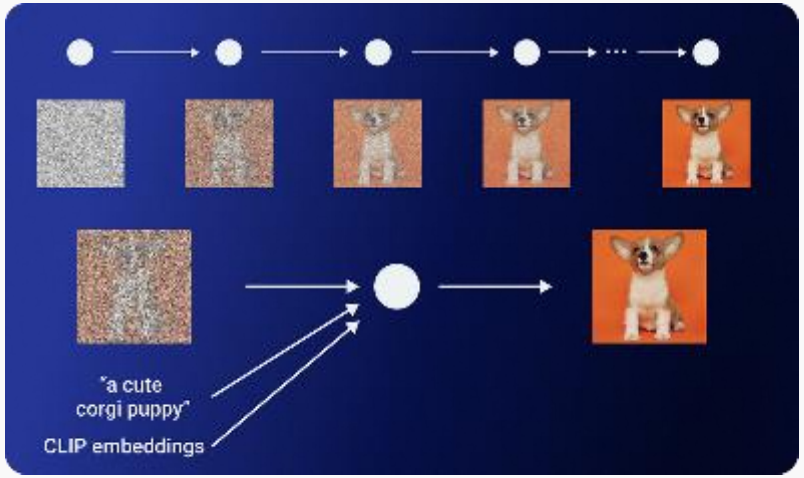
- It is a decoder only Transformer of width 2048 and 24 blocks, inputting:
    - Encoded text
    - CLIP text embedding
    - Embedding for the diffusion timestep
    - Noised CLIP image embedding
    - A final embedding whose output from the Transformer is used to predict the unnoised CLIP image embedding

DALL-E 2 Prior

- The idea is that we use a transformer to roughly translate and decode the text embedding into the image embedding.
- Unlike in typical stable diffusion, the authors use a loss function to predict the unnoised image embedding directly as opposed to predicting the ϵ noise.
- At sampling time, they generate two candidate image embeddings, and take the one that is more aligned with the text embedding (maximum dot product between the two embeddings, which are vectors).

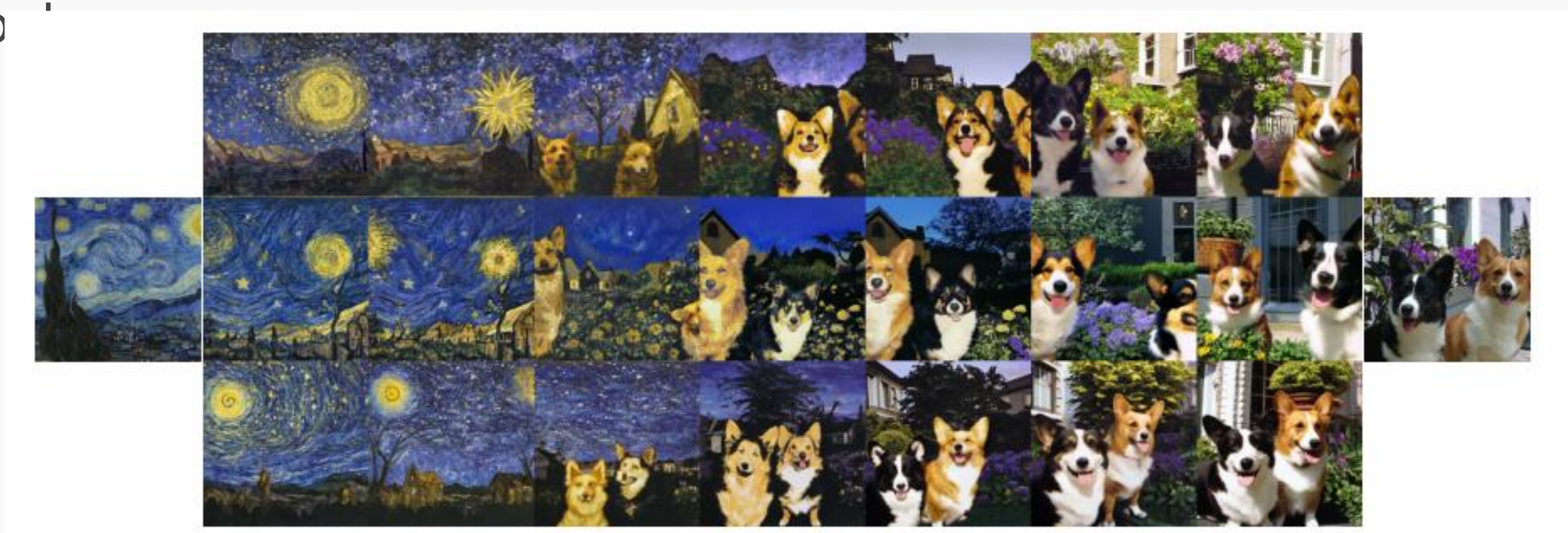
DALL-E 2 Decoder

- The decoder is a diffusion model. The goal of the decoder is to turn the image embeddings into actual images.
- They also optionally condition on the text captions in classifier-free guidance 50% of the time to reduce the variance of the outputs, and randomly setting the CLIP image embeddings to 0 about 10% of the time.

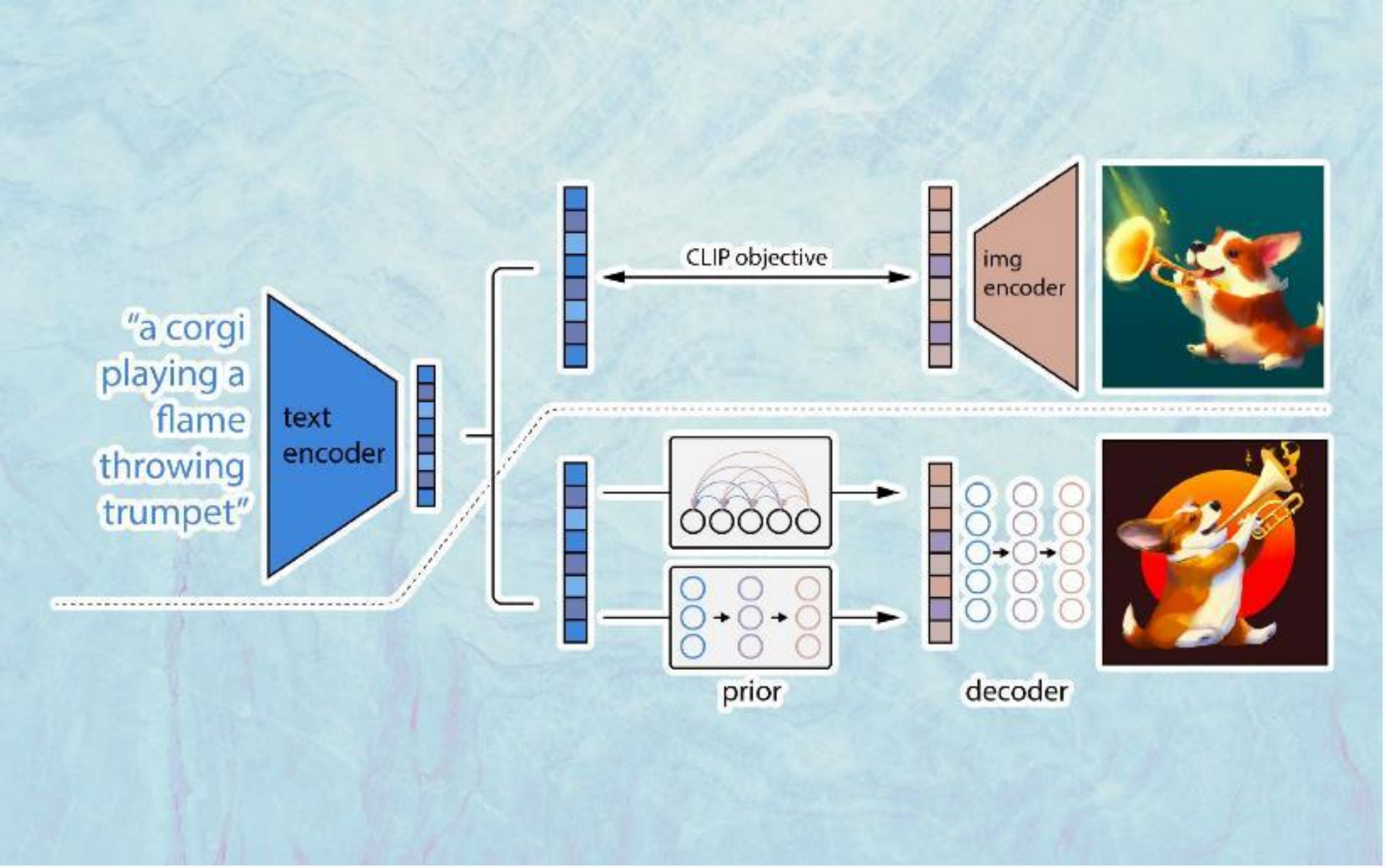


DALL-E 2 Decoder

- The decoder model allows users to create relevant images for any image embedding. As an example, the authors get two images and created image embeddings using CLIP.
- They then interpolated between these two embeddings and were able to generate images for these interpolated images as seen b



DALL-E 2


"a corgi playing a flame throwing trumpet"
text encoder
CLIP objective
img encoder
prior
decoder

That's all Folks!