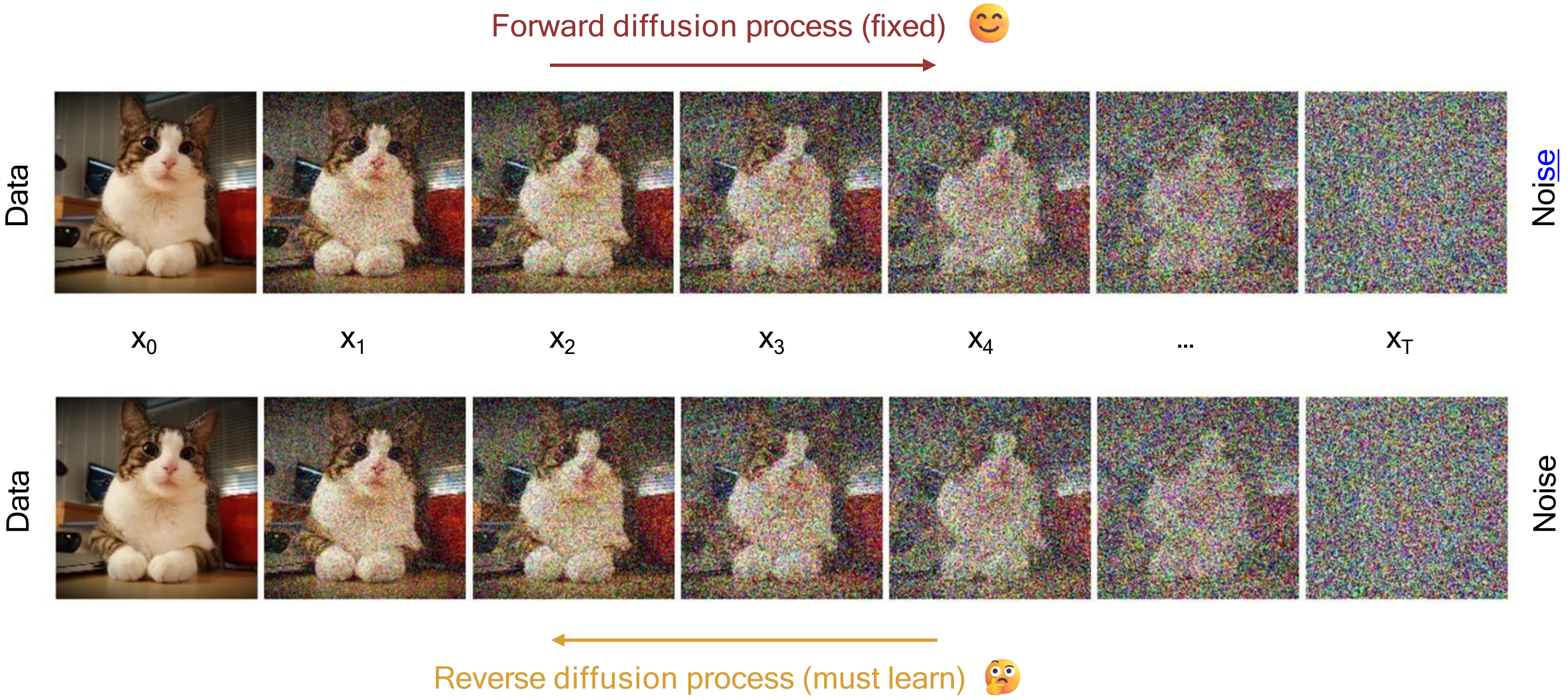
Diffusion Models

#DDPM Jonathan Ho, Ajay Jain, Pieter Abbeel. "Denoising Diffusion Probabilistic Models." NeurIPS, 2020.

Diffusion-based

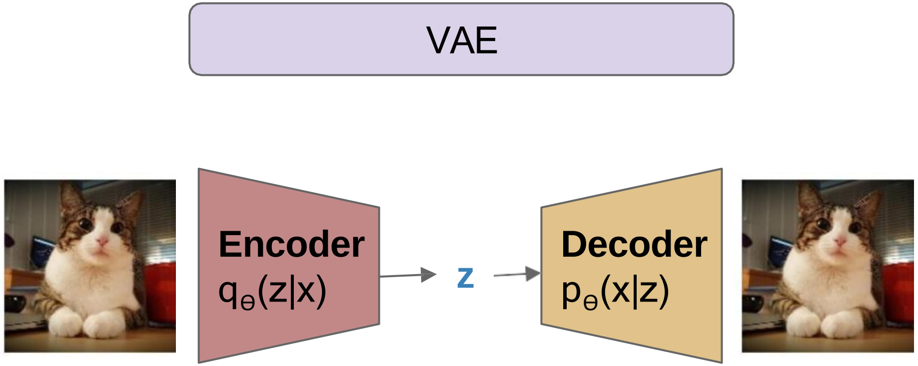


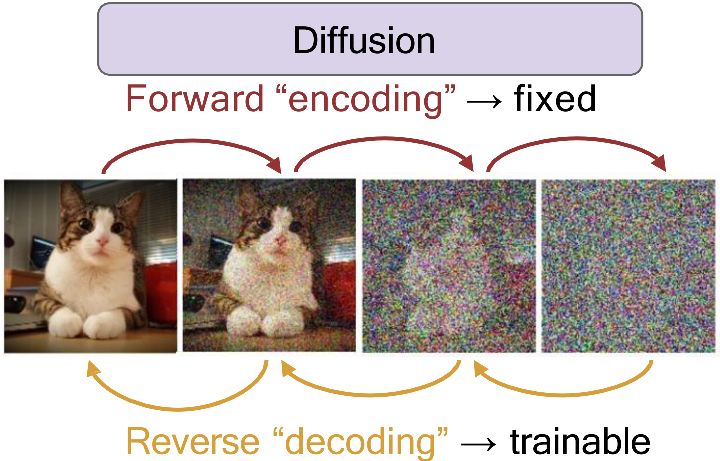
#**DDPM** Jonathan Ho, Ajay Jain, Pieter Abbeel. "Denoising Diffusion Probabilistic Models." NeurIPS, 2020.

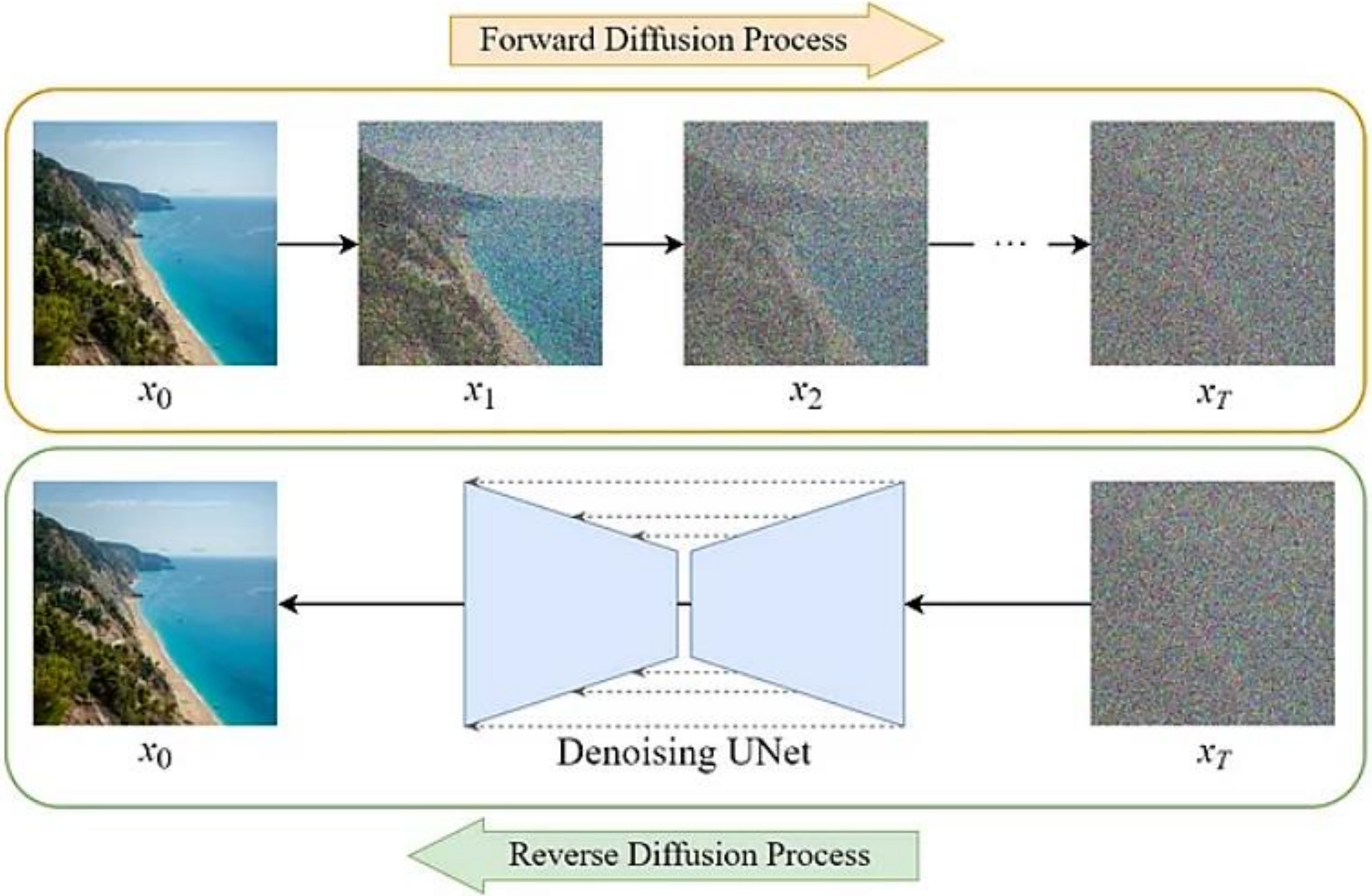
Relation with VAEs

Diffusion models can be considered as a special form of VAEs. However in diffusion models:

- The encoder is fixed: is **a predefined Markovian process**, meaning no learning happens here.
- The latent variables have the same dimension as the data: The noisy versions of the image at different timesteps retain the same size as the original image.
- The decoder is run multiple times in an autoregressive fashion: it runs in **multiple iterations** reconstructing the image from pure noise

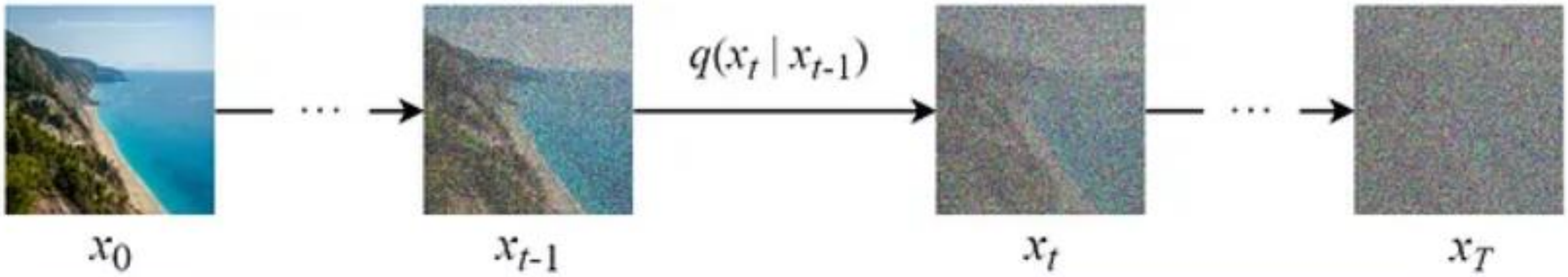






https://medium.com/@steinsfu/diffusion-model-clearly-explained-cd331bd41166#:~:text=The%20Diffusion%20model%20is%20divided,using%20a%20trained%20neural%20network.

Forward Diffusion Process



$$\overbrace{\boxed{q(x_t|x_{t-1})}}^{\text{Distribution of the noised images}} = \mathcal{N}(\overbrace{\boxed{x_t}}^{\text{Output}};\ \overbrace{\boxed{\sqrt{1-\beta_t}x_{t-1}}}^{\text{Mean } \mu_t},\ \overbrace{\boxed{\beta_t I}}^{\text{Variance } \Sigma_t})$$

Notations:

t : time step (from 1 to T)

x_0 : a data sampled from the real data distrition $q(x)$ (i.e. $x_0 \sim q(x)$)

β_t : variance schedule ($0 \leq \beta_t \leq 1$, and β_0 = small number, β_T = large number)

I : identity matrix

The forward diffusion process gradually adds Gaussian noise to the input image x_0 step by step, and there will be T steps in total. the process will produce a sequence of noisy image samples x_1, ..., x_T.

When T → ∞, the final result will become a completely noisy image as if it is sampled from an isotropic Gaussian distribution.

But instead of designing an algorithm to iteratively add noise to the image, we can use a closed-form formula to directly sample a noisy image at a specific time step t.

Ancestral Sampling (One Shot)

Ancestral sampling (One Shot)

$$x_t \sim p(x_t|x_0) = \mathcal{N}\left(x_t; \sqrt{\hat{\alpha}_t} \cdot x_0, (1-\hat{\alpha}_t)I\right)$$

$$\hat{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

$$\alpha_t = 1 - \beta_t$$

- Decompose the image generation (sampling) process in many small steps "denosing"
- Small step help to remove noise easily n accurately in backward process
- **Constant Variance:**
- Variance is kept constant, and mean is changed at each time stamp
- Changing both will collapse the structure of the image, and we need to reconstruct from noise.

How we get $\sqrt{1-\beta_t}$

Assume u start from a random variable x0 that follows a normal distribution

x $\sim \mathcal{N}(0,1)$

Noise Addition Formula in Diffusion Models

Step 1: Adding Noise

In the first step of the diffusion process ($t=1$), we scale the original image x_0 by a factor α and add Gaussian noise:

$$x_1 = \alpha x_0 + \sqrt{\beta_1}\epsilon_1$$

where:

- $\epsilon_1 \sim \mathcal{N}(0,1)$ is a random variable representing Gaussian noise with mean 0 and variance 1.

Calculating Variance

To compute the variance of x_1, we use the property that the variance of the sum of independent random variables is the sum of their variances:

$$\mathrm{Var}(x_1) = \mathrm{Var}(\alpha x_0 + \sqrt{\beta_1}\epsilon_1)$$

Expanding this:

$$\mathrm{Var}(x_1) = \alpha^2 \mathrm{Var}(x_0) + \beta_1 \mathrm{Var}(\epsilon_1)$$

Since:

- x_0 has unit variance: $\mathrm{Var}(x_0) = 1$,
- ϵ_1 also has unit variance: $\mathrm{Var}(\epsilon_1) = 1$,

  we get:

$$\mathrm{Var}(x_1) = \alpha^2 + \beta_1$$

Maintaining Unit Variance

To ensure that variance remains equal to 1 at every step, we enforce:

$$\alpha^2 + \beta_1 = 1$$

which gives:

$$\alpha_1 = \sqrt{1 - \beta_1}$$

Thus, the noise-added formulation can be rewritten as:

$$x_1 = \sqrt{1 - \beta_1} x_0 + \sqrt{\beta_1} \epsilon_1$$

$$x_1 = \alpha x_0 + \sqrt{\beta_1} \epsilon_1$$

This ensures that the variance remains normalized throughout the diffusion process.

Sample

$$x_t = \sqrt{1 - \beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_{t-1}$$

Distribution

$$\boxed{q(x_t | x_{t-1})} = \mathcal{N}(\boxed{x_t};\ \boxed{\sqrt{1 - \beta_t} x_{t-1}},\ \boxed{\beta_t I})$$

t=0 t=1 t=3 t=4 ... t=1000

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta}x_{t-1}, \beta_t I)$$

$$\rightarrow \quad q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

T=200

Closed form forward process

β_0=0.0001	β_1=0.0002	β_2=0.0003	β_3=0.0004	...	β_{200}=0.02
α_0=0.9999	α_1=0.9998	α_2=0.9997	α_3=0.9996		α_4=0.98
$\bar{\alpha}_0$=0.9999	$\overline{\alpha_1}$=0.9997	$\overline{\alpha_2}$=0.9994	$\overline{\alpha_3}$=0.9990		$\overline{\alpha_4}$=0.1322

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

Closed-Form Formula

The closed-form sampling formula can be derived using the Reparameterization Trick.

$$\text{If } \; z \sim \mathcal{N}(\mu, \sigma^2) \; \text{ then}$$
$$z = \mu + \sigma\varepsilon \qquad \text{where } \; \varepsilon \sim \mathcal{N}(0, 1)$$

Reparameterization trick

With this trick, we can express the sampled image x_t as follows:

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_{t-1}$$

Express $\mathbf{x}_t$ using the reparameterization trick

Then we can expand it recursively to get the closed-form formula:

$$
\begin{aligned}
x_t &= \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_{t-1} \\
&= \sqrt{\alpha_t}\, \boxed{x_{t-1}} + \sqrt{1-\alpha_t}\, \varepsilon_{t-1} \\
&= \sqrt{\alpha_t}\, \boxed{\left(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \varepsilon_{t-2}\right)} + \sqrt{1-\alpha_t}\, \varepsilon_{t-1} \\
&= \sqrt{\alpha_t\, \alpha_{t-1}}\, x_{t-2} + \boxed{\sqrt{\alpha_t(1-\alpha_{t-1})}\, \varepsilon_{t-2} + \sqrt{1-\alpha_t}\, \varepsilon_{t-1}} \\
&= \sqrt{\alpha_t\, \alpha_{t-1}}\, x_{t-2} + \boxed{\sqrt{1-\alpha_t\, \alpha_{t-1}}\, \bar{\varepsilon}_{t-2}} \quad \text{How?} \\
&\;\;\vdots \\
&= \sqrt{\alpha_t\, \alpha_{t-1}\, \ldots\, \alpha_1}\, x_0 + \sqrt{1-\alpha_t\, \alpha_{t-1}\, \ldots\, \alpha_1}\, \varepsilon \\
&= \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
&\varepsilon_0, \ldots, \varepsilon_{t-2}, \varepsilon_{t-1} \sim \mathcal{N}(0, I) \\
&\bar{\varepsilon}_0, \ldots, \bar{\varepsilon}_{t-2}, \bar{\varepsilon}_{t-1} \sim \mathcal{N}(0, I) \\
&\varepsilon \sim \mathcal{N}(0, I) \\
&\alpha_t = 1 - \beta_t \\
&\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i
\end{aligned}
$$

Derivation of the closed-form formula

But how do we jump from line 4 to line 5?

$$\boxed{\sqrt{\alpha_t(1-\alpha_{t-1})}\,\varepsilon_{t-2} + \sqrt{1-\alpha_t}\,\varepsilon_{t-1}}$$

$$\boxed{\sqrt{1-\alpha_t\,\alpha_{t-1}}\,\bar{\varepsilon}_{t-2}}$$

How?

$$
\begin{aligned}
x_t &= \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_{t-1} \\
&= \sqrt{\alpha_t}\, \boxed{x_{t-1}} + \sqrt{1-\alpha_t}\, \varepsilon_{t-1} \\
&= \sqrt{\alpha_t}\, \boxed{\left(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \varepsilon_{t-2}\right)} + \sqrt{1-\alpha_t}\, \varepsilon_{t-1} \\
&= \sqrt{\alpha_t\, \alpha_{t-1}}\, x_{t-2} + \boxed{\sqrt{\alpha_t(1-\alpha_{t-1})}\, \varepsilon_{t-2}} + \boxed{\sqrt{1-\alpha_t}\, \varepsilon_{t-1}}
\end{aligned}
$$

$$\varepsilon_0, \ldots, \varepsilon_{t-2}, \varepsilon_{t-1} \sim \mathcal{N}(0, I)$$

$$\bar{\varepsilon}_0, \ldots, \bar{\varepsilon}_{t-2}, \bar{\varepsilon}_{t-1} \sim \mathcal{N}(0, I)$$

$$\varepsilon \sim \mathcal{N}(0, I)$$

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

$$X + Y$$

Reparameterization Trick		Underlying Normal Distribution
$0 + \sqrt{\alpha_t(1-\alpha_{t-1})}\, \varepsilon_{t-2}$	$\dashrightarrow$	$X \sim \mathcal{N}(0,\ \alpha_t(1-\alpha_{t-1})\, I)$
$0 + \sqrt{1-\alpha_t}\, \varepsilon_{t-1}$	$\dashrightarrow$	$Y \sim \mathcal{N}(0,\ (1-\alpha_t)\, I)$

Recall:

If $X \sim \mathcal{N}(\mu_X, \sigma_X^2) \quad Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$

$Z = X + Y$

Then $Z \sim \mathcal{N}(\mu_X + \mu_Y,\ \sigma_X^2 + \sigma_Y^2)$

$$Z \sim \mathcal{N}(0, \boxed{\sigma_X^2 + \sigma_Y^2})$$

$$Z \sim \mathcal{N}(0, \boxed{(1 - \alpha_t \alpha_{t-1})}\, I)$$

$$\begin{aligned} \sigma_X^2 + \sigma_Y^2 &= \alpha_t(1 - \alpha_{t-1}) + (1 - \alpha_t) \\ &= \cancel{\alpha_t} - \alpha_t \alpha_{t-1} + 1 - \cancel{\alpha_t} \\ &= 1 - \alpha_t \alpha_{t-1} \end{aligned}$$

Reparameterization Trick

$$0 + \sqrt{1 - \alpha_t \alpha_{t-1}}\, \bar{\varepsilon}_{t-2}$$

$$= \sqrt{\alpha_t\, \alpha_{t-1}}\, x_{t-2} + \boxed{\sqrt{1 - \alpha_t\, \alpha_{t-1}}\, \bar{\varepsilon}_{t-2}}$$

$$\vdots$$

$$= \sqrt{\alpha_t\, \alpha_{t-1}\, \ldots\, \alpha_1}\, x_0 + \sqrt{1 - \alpha_t\, \alpha_{t-1}\, \ldots\, \alpha_1}\, \varepsilon$$

$$= \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$$

Detailed derivation from line 4 to line 5

Repeating these steps will give us the following formula which depends only on the input image x_0:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$$

The closed-form formula

Now we can directly sample x_t at any time step using this formula, and this makes the forward process much faster.

where:

- x_0 is the original clean image,
- x_t is the noisy image at timestep t,
- $\bar{\alpha}_t$ is the accumulated product of all previous noise scales,
- $\epsilon \sim \mathcal{N}(0, I)$ is standard Gaussian noise.

To extract ϵ_t (the noise at step t) from a given noisy sample x_t:

$$\epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{\sqrt{1 - \bar{\alpha}_t}}$$

How Do We Compute $\bar{\alpha}_t$?

$\bar{\alpha}_t$ is the cumulative product of a sequence of predefined noise scales α_t. These α_t values are usually determined by a noise schedule, which is a function that controls how noise is added across timesteps. The most common schedules include:

- **Linear schedule:** β_t increases linearly from a small to a large value.
- **Cosine schedule**: Inspired by cosine functions for smoother noise distribution.

Once we have a noise schedule, we compute:

$$\alpha_t = 1 - \beta_t$$

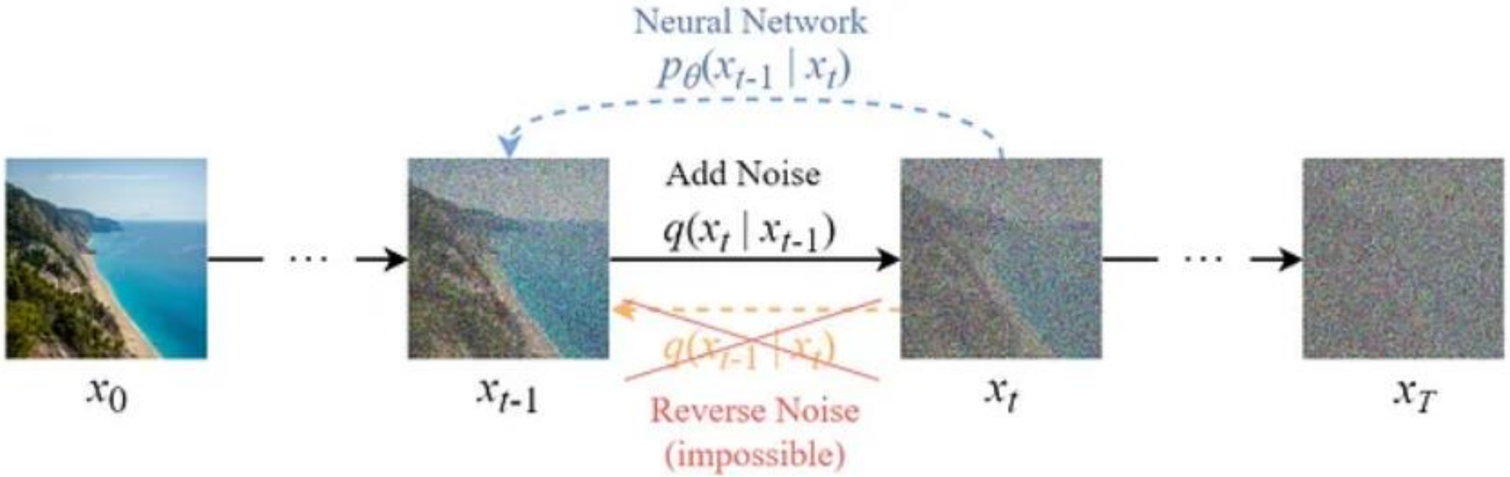
$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

the noise schedule provides all the α_t values upfront, allowing you to calculate any $\bar{\alpha}_t$ directly without needing to step through the diffusion process.

This cumulative product accumulates the effect of all past timesteps.

If you only have the original image x_0, you must define a noise schedule first (e.g., a linear or cosine schedule). Then, compute $\bar{\alpha}_t$ as the cumulative product of α_t up to timestep t. This allows you to generate noisy samples in one step using the equation.

Reverse Diffusion Process



Target Distribution $$q(x_{t-1}|x_t) = \mathcal{N}(x_{t-1};\ \tilde{\mu}_t(x_t, x_0),\ \tilde{\beta}_t I)$$

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

Learnable parameters
(Neural Network)

Unlike the forward process, we cannot use $q(x_{t-1}|x_t)$ to reverse the noise since it is intractable (uncomputable).

Thus we need to train a neural network $p\theta(x_{t-1}|x_t)$ to approximate $q(x_{t-1}|x_t)$. The approximation $p\theta(x_{t-1}|x_t)$ follows a normal distribution and its mean and variance are set as follows:

$$\begin{cases} \mu_\theta(x_t, t) & := & \tilde{\mu}_t(x_t, x_0) \\ \Sigma_\theta(x_t, t) & := & \tilde{\beta}_t I \end{cases}$$

mean and variance of **pθ**

Loss Function

We can define our loss as a Negative Log-Likelihood:

$$\text{Loss} = -\log(\boxed{p_\theta(x_0)})$$

Depends on $x_1, x_2, \ldots, x_T$
Therefore it is intractable!

Negative log-likelihood

This setup is very similar to the one in VAE. instead of optimizing the intractable loss function itself, we can optimize the Variational Lower Bound.

To approximate the target denoising step q, we only need to approximate its mean using a neural network. So we set the approximated mean $\mu\theta$ to be in the same form as the target mean $\tilde{\mu_t}$ (with a learnable neural network $\varepsilon\theta$):

$$\tilde{\boldsymbol{\mu}}_t(x_t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boxed{\boldsymbol{\varepsilon}_t}\right)$$

$$\text{Set} \quad \boldsymbol{\mu}_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boxed{\boldsymbol{\varepsilon}_\theta(x_t, t)}\right)$$

Approximated mean

Simplified Loss

So the final simplified training objective is as follows:

$$\boxed{x_t = \sqrt{\bar{a}_t}\, x_0 + \sqrt{1 - \bar{a}_t}\, \varepsilon}$$

$$L_{\text{simple}} = \mathbb{E}_{t, x_0, \boldsymbol{\varepsilon}}\left[\left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\boxed{x_t}, t) \right\|^2 \right]$$

Simplified training objective

Loss Function: The UNet is trained to minimize the difference between its predicted noise $\hat{\epsilon}_\theta$ and the actual noise ϵ (which was added during forward diffusion):

$$L = \mathbb{E}_{x_0, \epsilon, t}\left[\|\epsilon - \hat{\epsilon}_\theta(x_t, t)\|^2 \right]$$

Reverse inference/sampling part

1. Input Noisy Image x_t

- Given an image x_t at time t, we want to move backward in the diffusion process to reconstruct x_{t-1}.

2. UNet Predicts Noise $\epsilon_\theta(x_t, t)$

- The model (usually a **UNet**) takes in x_t and estimates the noise component $\epsilon_\theta(x_t, t)$ that was added in the forward process.

3. Estimate the Original Image x_0 (Intermediate Step)

- Using the closed-form **forward process equation**, we estimate the **original clean image** x_0:

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}$$

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$$

- This equation removes the predicted noise from x_t and estimates what the original image might have been before the noise was added.

4. Compute the Mean $\mu_\theta(x_t, t)$ for Transition to x_{t-1}

- Now that we have an estimate of x_0, we compute the mean for the next step:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boxed{\epsilon_\theta(x_t, t)}\right)$$

- This tells us where x_{t-1} should be centered.

5. Sample x_{t-1} Using a Stochastic Term

- Instead of directly setting $x_{t-1} = \mu_\theta(x_t, t)$, we **add Gaussian noise** to make the process probabilistic:

$$x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z, \quad \text{where } z \sim \mathcal{N}(0, I)$$

- σ_t is a predefined noise scale (depends on the variance schedule).
- z is standard Gaussian noise, ensuring some randomness in the process.

6. Repeat Until x_0 is Reached

- This step-by-step denoising process continues until we reach x_0, the final clean image.

Why do we add noise when denoising?

•The forward diffusion process was **stochastic** (random noise added at each step).
•To reverse it, we also need a **stochastic process** rather than a deterministic one.
•The added noise helps maintain diversity in generated images.

•**Creativity and Variation:** It allows the model to generate creative and varied outputs, even when starting from the same latent noise. Imagine generating different variations of a "cat" image – different poses, colors, fur patterns, etc.
•**Avoiding Overfitting:** Stochasticity helps prevent the model from memorizing specific training examples and encourages it to learn the underlying distribution of the data.
•**Exploring the Latent Space:** The added noise allows the reverse process to explore different regions of the latent space, leading to a wider range of generated samples.

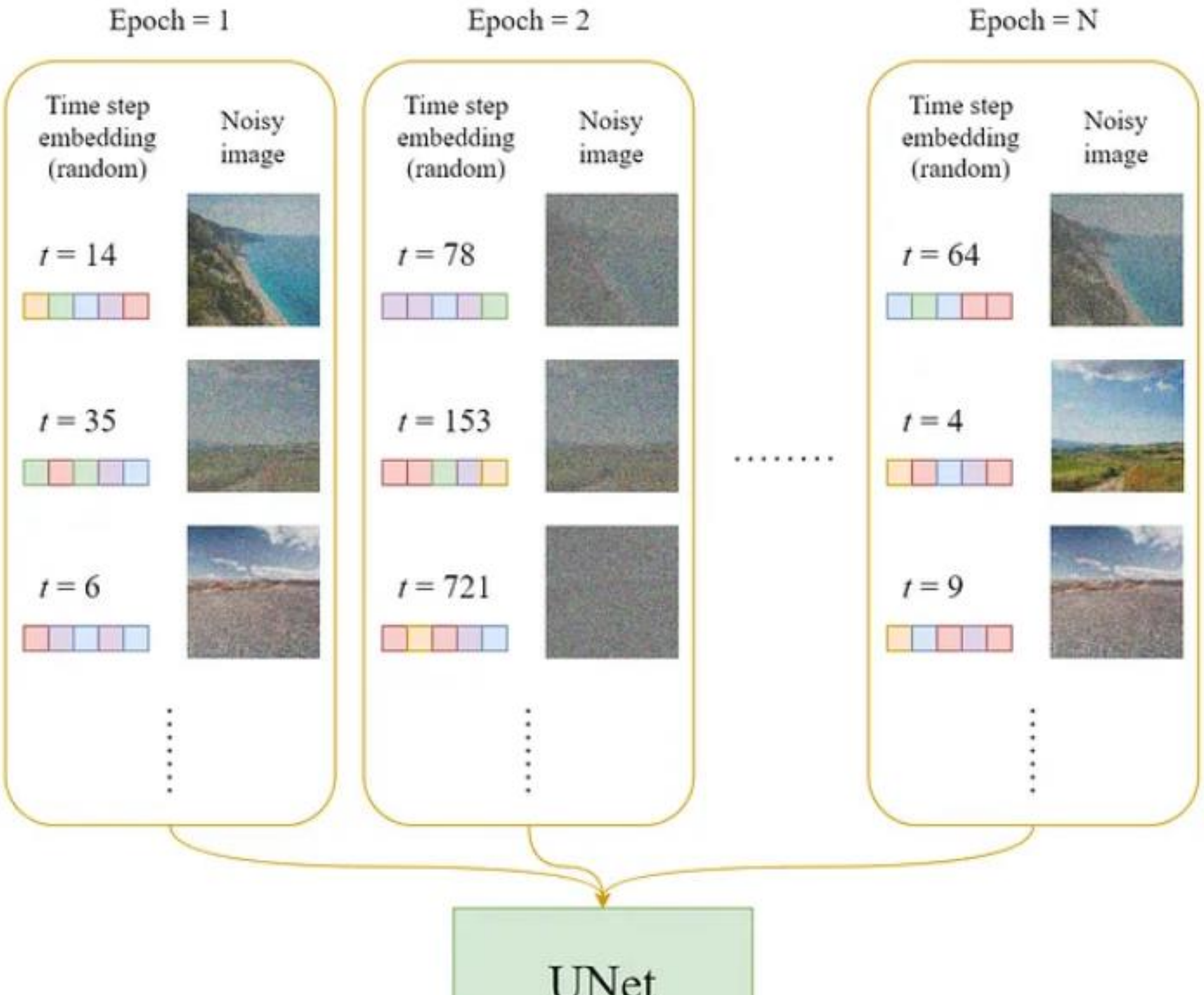
The U-Net Model

Dataset

In each epoch:

1. A random time step t will be selected for each training sample (image).
2. Apply the Gaussian noise (corresponding to t) to each image.
3. Convert the time steps to embeddings (vectors).

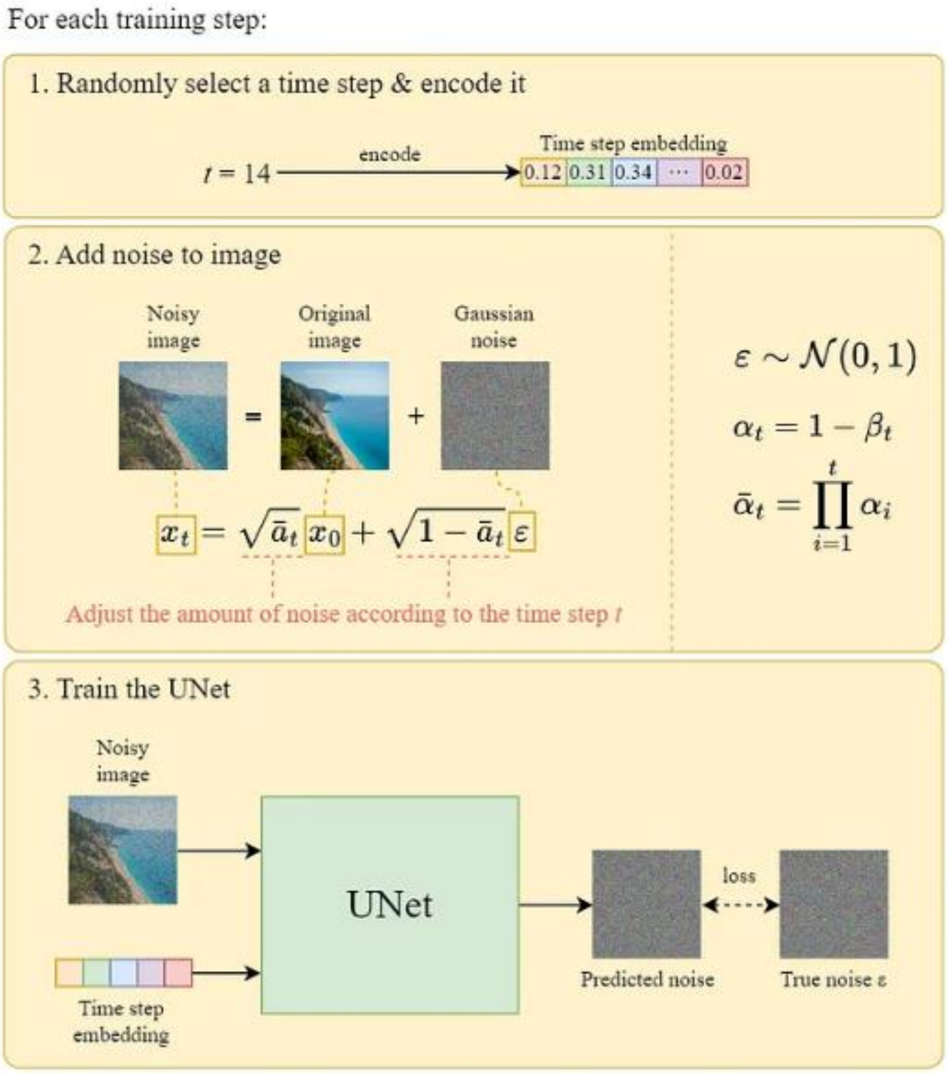
Why do we pass random time steps in Uet during training?



Algorithm 1 Training

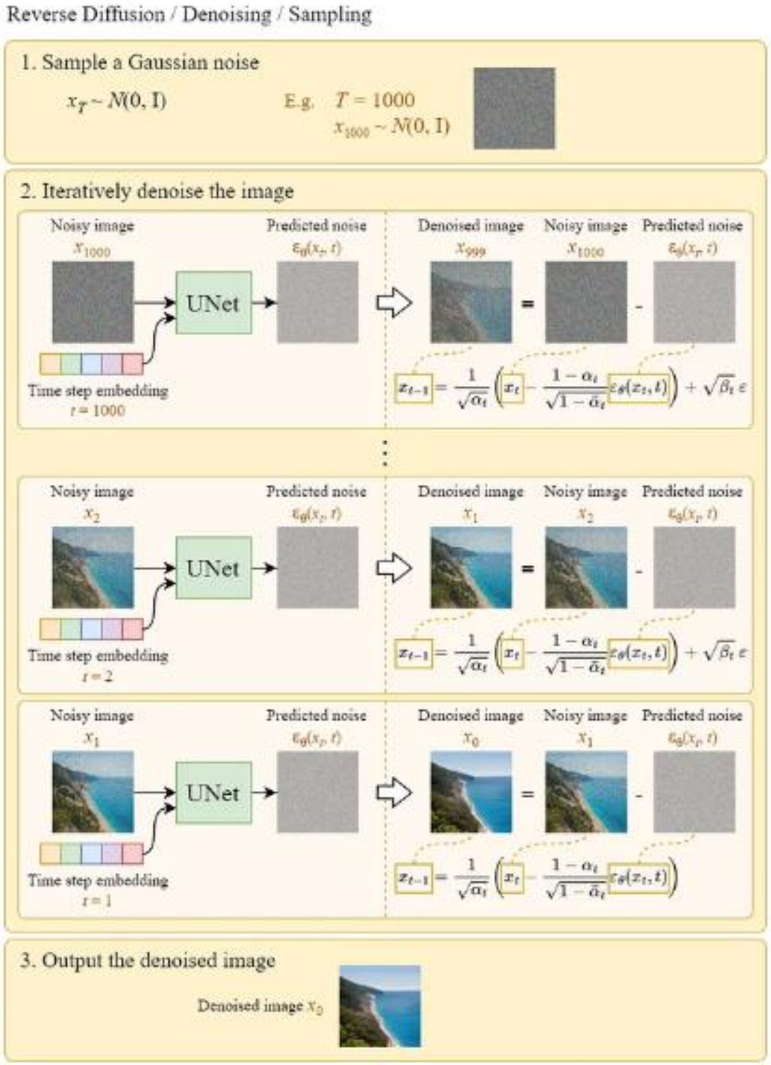
repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

During inference, the forward process is skipped—we directly sample noise and denoise step by step.



Algorithm 2 Sampling

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

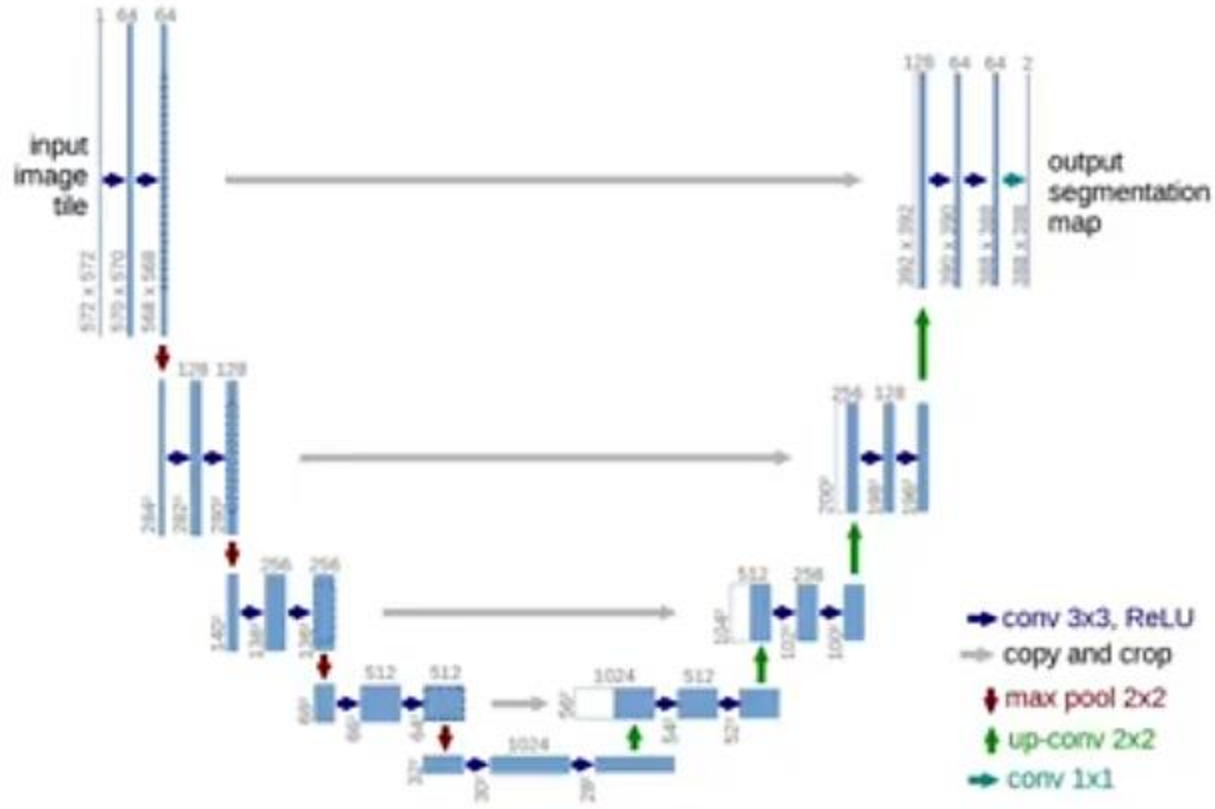


Summary

Here are some main takeaways from this article:

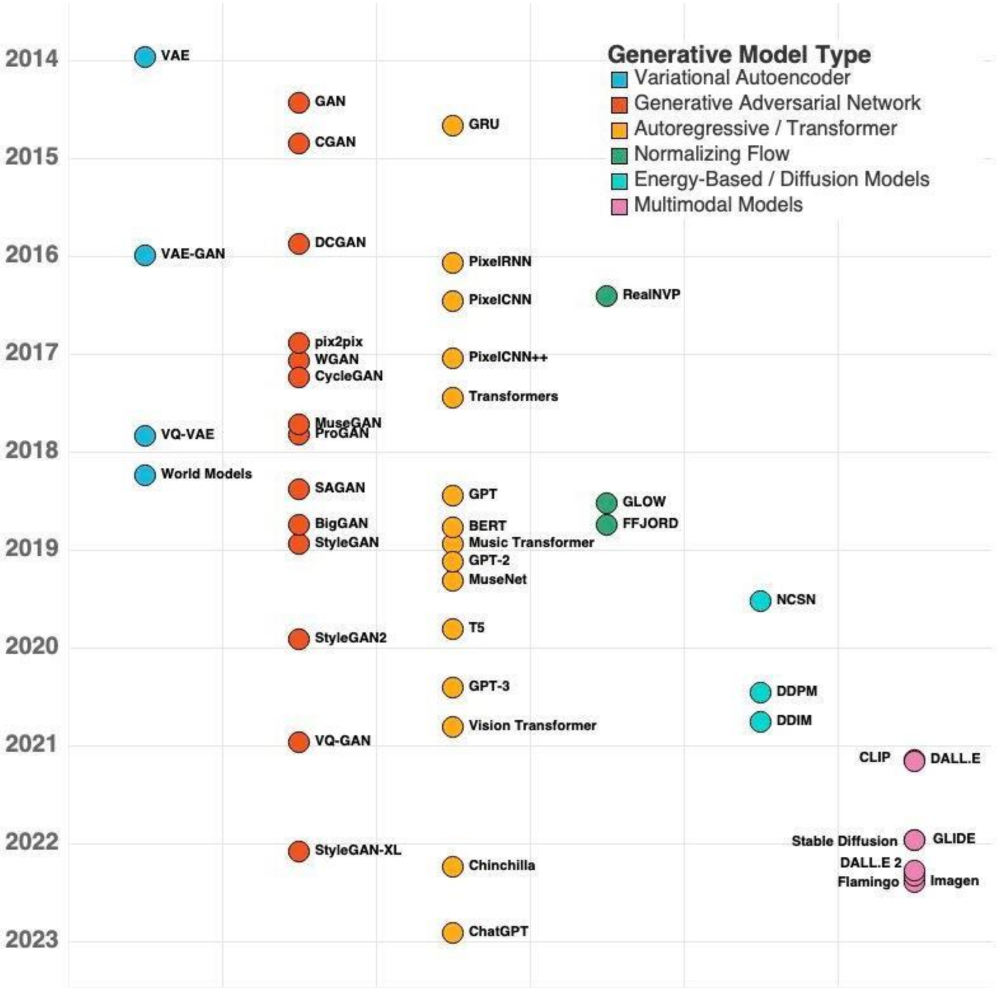
- The Diffusion model is divided into two parts: forward diffusion and reverse diffusion.
- The forward diffusion can be done using the closed-form formula.
- The backward diffusion can be done using a trained neural network.
- To approximate the desired denoising step q, we just need to approximate the noise ε_t using a neural network $\varepsilon\theta$.
- Training on the simplified loss function yields better sample quality.

UNET model (architecture)



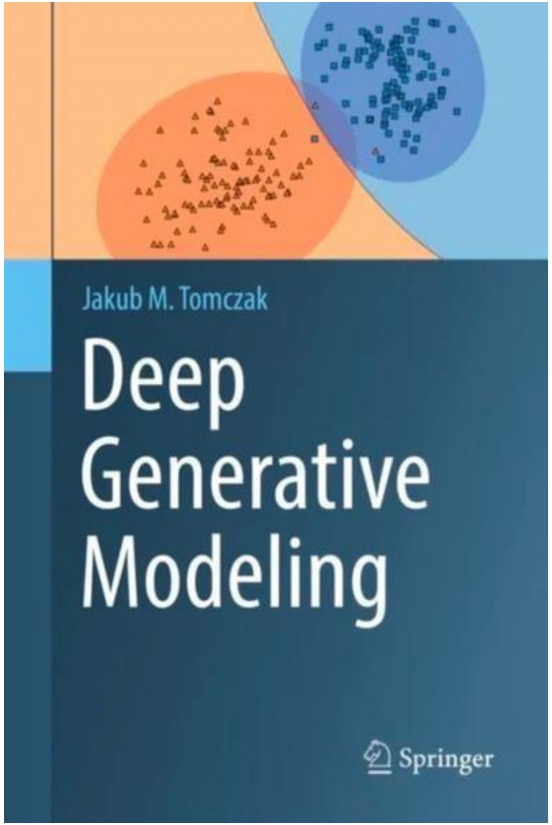
https://github.com/hkproj/pytorch-ddpm

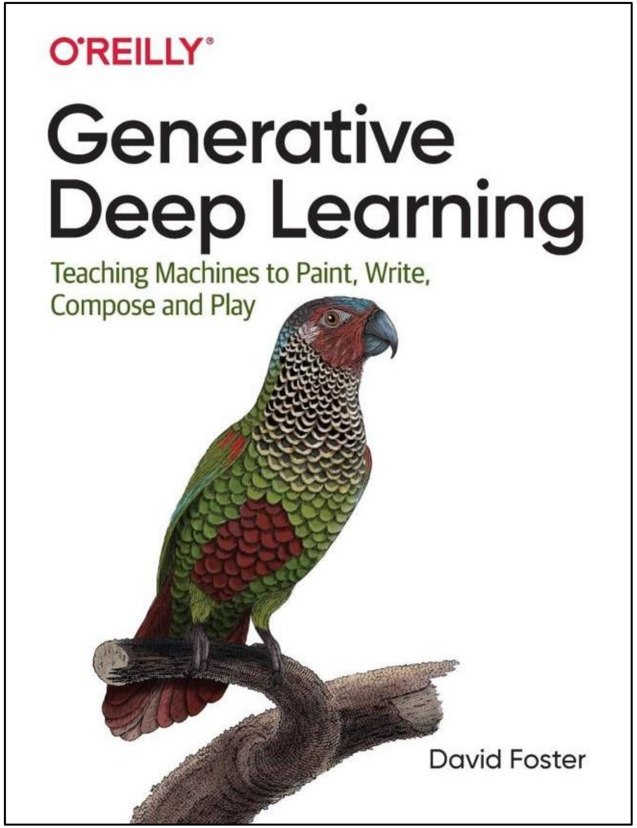
Generative AI Timeline



Source: [David Foster]

Recommended books





Interview of David Foster for Machine Learning Street Talk (2023)