Diffusion model

In machine learning, **diffusion models**, also known as **diffusion probabilistic models** or **score-based generative models**, are a class of latent variable generative models. A diffusion model consists of three major components: the forward process, the reverse process, and the sampling procedure.[1] The goal of diffusion models is to learn a diffusion process for a given dataset, such that the process can generate new elements that are distributed similarly as the original dataset. A diffusion model models data as generated by a diffusion process, whereby a new datum performs a random walk with drift through the space of all possible data.[2] A trained diffusion model can be sampled in many ways, with different efficiency and quality.

There are various equivalent formalisms, including Markov chains, denoising diffusion probabilistic models, noise conditioned score networks, and stochastic differential equations.[3] They are typically trained using variational inference.[4] The model responsible for denoising is typically called its "backbone". The backbone may be of any kind, but they are typically U-nets or transformers.

As of 2024, diffusion models are mainly used for computer vision tasks, including image denoising, inpainting, super-resolution, image generation, and video generation. These typically involve training a neural network to sequentially denoise images blurred with Gaussian noise.[2][5] The model is trained to reverse the process of adding noise to an image. After training to convergence, it can be used for image generation by starting with an image composed of random noise, and applying the network iteratively to denoise the image.

Diffusion-based image generators have seen widespread commercial interest, such as Stable Diffusion and DALL-E. These models typically combine diffusion models with other models, such as text-encoders and cross-attention modules to allow text-conditioned generation.[6]

Other than computer vision, diffusion models have also found applications in natural language processing[7][8] such as text generation[9][10] and summarization,[11] sound generation,[12] and reinforcement learning.[13][14]

Denoising diffusion model

Non-equilibrium thermodynamics

Diffusion models were introduced in 2015 as a method to train a model that can sample from a highly complex probability distribution. They used techniques from non-equilibrium thermodynamics, especially diffusion.[15]

Consider, for example, how one might model the distribution of all naturally-occurring photos. Each image is a point in the space of all images, and the distribution of naturally-occurring photos is a "cloud" in space, which, by repeatedly adding noise to the images, diffuses out to the rest of the image space, until the cloud becomes all but indistinguishable from a Gaussian distribution $\mathcal{N}(0, I)$. A model that can approximately undo the diffusion can then be used to sample from the original distribution. This is studied in "non-equilibrium" thermodynamics, as the starting distribution is not in equilibrium, unlike the final distribution.

The equilibrium distribution is the Gaussian distribution $\mathcal{N}(0, I)$, with pdf $\rho(x) \propto e^{-\frac{1}{2}\|x\|^2}$. This is just the Maxwell–Boltzmann distribution of particles in a potential well $V(x) = \frac{1}{2}\|x\|^2$ at temperature 1. The initial distribution, being very much out of equilibrium, would diffuse towards the equilibrium distribution, making biased random steps that are a sum of pure randomness (like a Brownian walker) and gradient descent down the potential well. The randomness is necessary: if the particles were to undergo only gradient descent, then they will all fall to the origin, collapsing the distribution.

Denoising Diffusion Probabilistic Model (DDPM)

The 2020 paper proposed the Denoising Diffusion Probabilistic Model (DDPM), which improves upon the previous method by variational inference.[4][16]

Forward diffusion

To present the model, we need some notation.

- $\beta_1, \dots, \beta_T \in (0, 1)$ are fixed constants.
- $\alpha_t := 1 - \beta_t$
- $\bar{\alpha}_t := \alpha_1 \cdots \alpha_t$
- $\sigma_t := \sqrt{1 - \bar{\alpha}_t}$
- $\tilde{\sigma}_t := \frac{\sigma_{t-1}}{\sigma_t}\sqrt{\beta_t}$
- $\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)x_0}{\sigma_t^2}$
- $\mathcal{N}(\mu, \Sigma)$ is the normal distribution with mean μ and variance Σ, and $\mathcal{N}(x|\mu, \Sigma)$ is the probability density at x.
- A vertical bar denotes conditioning.

A **forward diffusion process** starts at some starting point $x_0 \sim q$, where q is the probability distribution to be learned, then repeatedly adds noise to it by

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

where $z_1, \dots, z_T$ are IID samples from $\mathcal{N}(0, I)$. This is designed so that for any starting distribution of x_0, we have $\lim_t x_t|x_0$ converging to $\mathcal{N}(0, I)$.

The entire diffusion process then satisfies

$$q(x_{0:T}) = q(x_0)q(x_1|x_0)\cdots q(x_T|x_{T-1}) = q(x_0)\mathcal{N}(x_1|\sqrt{\alpha_1}x_0, \beta_1 I)\cdots\mathcal{N}(x_T|\sqrt{\alpha_T}x_{T-1}, \beta_T I)$$

or

$$\ln q(x_{0:T}) = \ln q(x_0) - \sum_{t=1}^{T}\frac{1}{2\beta_t}\|x_t - \sqrt{1 - \beta_t}x_{t-1}\|^2 + C$$

where C is a normalization constant and often omitted. In particular, we note that $x_{1:T}|x_0$ is a gaussian process, which affords us considerable freedom in reparameterization. For example, by standard manipulation with gaussian process,

$$x_t|x_0 \sim N\left(\sqrt{\bar\alpha_t}x_0, \sigma_t^2 I\right)$$

$$x_{t-1}|x_t, x_0 \sim \mathcal{N}(\tilde\mu_t(x_t, x_0), \tilde\sigma_t^2 I)$$

In particular, notice that for large t, the variable $x_t|x_0 \sim N\left(\sqrt{\bar\alpha_t}x_0, \sigma_t^2 I\right)$ converges to $\mathcal{N}(0, I)$. That is, after a long enough diffusion process, we end up with some x_T that is very close to $\mathcal{N}(0, I)$, with all traces of the original $x_0 \sim q$ gone.

For example, since

$$x_t|x_0 \sim N\left(\sqrt{\bar\alpha_t}x_0, \sigma_t^2 I\right)$$

we can sample $x_t|x_0$ directly "in one step", instead of going through all the intermediate steps $x_1, x_2, \ldots, x_{t-1}$.

Derivation by reparameterization

We know $x_{t-1}|x_0$ is a gaussian, and $x_t|x_{t-1}$ is another gaussian. We also know that these are independent. Thus we can perform a reparameterization:

$$x_{t-1} = \sqrt{\bar\alpha_{t-1}}x_0 + \sqrt{1-\bar\alpha_{t-1}}z$$

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}z'$$

where z, z' are IID gaussians.

There are 5 variables x_0, x_{t-1}, x_t, z, z' and two linear equations. The two sources of randomness are z, z', which can be reparameterized by rotation, since the IID gaussian distribution is rotationally symmetric.

By plugging in the equations, we can solve for the first reparameterization:

$$x_t = \sqrt{\bar\alpha_t}x_0 + \underbrace{\sqrt{\alpha_t - \bar\alpha_t}z + \sqrt{1-\alpha_t}z'}_{=\sigma_t z''}$$

where z'' is a gaussian with mean zero and variance one.

To find the second one, we complete the rotational matrix:

$$\begin{bmatrix} z'' \\ z''' \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\alpha_t - \bar\alpha_t}}{\sigma_t} & \frac{\sqrt{\beta_t}}{\sigma_t} \\ ? & ? \end{bmatrix} \begin{bmatrix} z \\ z' \end{bmatrix}$$

Since rotational matrices are all of the form $\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$, we know the matrix must be

$$\begin{bmatrix} z'' \\ z''' \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\alpha_t - \bar\alpha_t}}{\sigma_t} & \frac{\sqrt{\beta_t}}{\sigma_t} \\ -\frac{\sqrt{\beta_t}}{\sigma_t} & \frac{\sqrt{\alpha_t - \bar\alpha_t}}{\sigma_t} \end{bmatrix} \begin{bmatrix} z \\ z' \end{bmatrix}$$

and since the inverse of rotational matrix is its transpose,

$$\begin{bmatrix} z \\ z' \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\alpha_t - \bar{\alpha}_t}}{\sigma_t} & -\frac{\sqrt{\beta_t}}{\sigma_t} \\ \frac{\sqrt{\beta_t}}{\sigma_t} & \frac{\sqrt{\alpha_t - \bar{\alpha}_t}}{\sigma_t} \end{bmatrix} \begin{bmatrix} z'' \\ z''' \end{bmatrix}$$

Plugging back, and simplifying, we have

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sigma_t z''$$
$$x_{t-1} = \tilde{\mu}_t(x_t, x_0) - \tilde{\sigma}_t z'''$$

Backward diffusion

The key idea of DDPM is to use a neural network parametrized by θ. The network takes in two arguments x_t, t, and outputs a vector $\mu_\theta(x_t, t)$ and a matrix $\Sigma_\theta(x_t, t)$, such that each step in the forward diffusion process can be approximately undone by $x_{t-1} \sim \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. This then gives us a backward diffusion process p_θ defined by

$$p_\theta(x_T) = \mathcal{N}(x_T | 0, I)$$
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

The goal now is to learn the parameters such that $p_\theta(x_0)$ is as close to $q(x_0)$ as possible. To do that, we use maximum likelihood estimation with variational inference.

Variational inference

The ELBO inequality states that $\ln p_\theta(x_0) \geq E_{x_{1:T} \sim q(\cdot|x_0)}[\ln p_\theta(x_{0:T}) - \ln q(x_{1:T}|x_0)]$, and taking one more expectation, we get

$$E_{x_0 \sim q}[\ln p_\theta(x_0)] \geq E_{x_{0:T} \sim q}[\ln p_\theta(x_{0:T}) - \ln q(x_{1:T}|x_0)]$$

We see that maximizing the quantity on the right would give us a lower bound on the likelihood of observed data. This allows us to perform variational inference.

Define the loss function

$$L(\theta) := -E_{x_{0:T} \sim q}[\ln p_\theta(x_{0:T}) - \ln q(x_{1:T}|x_0)]$$

and now the goal is to minimize the loss by stochastic gradient descent. The expression may be simplified to[17]

$$L(\theta) = \sum_{t=1}^{T} E_{x_{t-1}, x_t \sim q}[-\ln p_\theta(x_{t-1}|x_t)] + E_{x_0 \sim q}[D_{KL}(q(x_T|x_0) \| p_\theta(x_T))] + C$$

where C does not depend on the parameter, and thus can be ignored. Since $p_\theta(x_T) = \mathcal{N}(x_T|0, I)$ also does not depend on the parameter, the term $E_{x_0 \sim q}[D_{KL}(q(x_T|x_0) \| p_\theta(x_T))]$ can also be ignored. This leaves just $L(\theta) = \sum_{t=1}^{T} L_t$ with $L_t = E_{x_{t-1}, x_t \sim q}[-\ln p_\theta(x_{t-1}|x_t)]$ to be minimized.

Noise prediction network

Since $x_{t-1}|x_t, x_0 \sim \mathcal{N}(\tilde{\mu}_t(x_t, x_0), \tilde{\sigma}_t^2 I)$, this suggests that we should use $\mu_\theta(x_t, t) = \tilde{\mu}_t(x_t, x_0)$; however, the network does not have access to x_0, and so it has to estimate it instead. Now, since $x_t|x_0 \sim N\left(\sqrt{\bar{\alpha}_t}x_0, \sigma_t^2 I\right)$, we may write $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sigma_t z$, where z is some unknown gaussian noise. Now we see that estimating x_0 is equivalent to estimating z.

Therefore, let the network output a noise vector $\epsilon_\theta(x_t, t)$, and let it predict

$$\mu_\theta(x_t, t) = \tilde{\mu}_t\left(x_t, \frac{x_t - \sigma_t \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}\right) = \frac{x_t - \epsilon_\theta(x_t, t)\beta_t/\sigma_t}{\sqrt{\alpha_t}}$$

It remains to design $\Sigma_\theta(x_t, t)$. The DDPM paper suggested not learning it (since it resulted in "unstable training and poorer sample quality"), but fixing it at some value $\Sigma_\theta(x_t, t) = \zeta_t^2 I$, where either $\zeta_t^2 = \beta_t$ or $\tilde{\sigma}_t^2$ yielded similar performance.

With this, the loss simplifies to

$$L_t = \frac{\beta_t^2}{2\alpha_t \sigma_t^2 \zeta_t^2} E_{x_0 \sim q; z \sim \mathcal{N}(0, I)}\left[\left\|\epsilon_\theta(x_t, t) - z\right\|^2\right] + C$$

which may be minimized by stochastic gradient descent. The paper noted empirically that an even simpler loss function

$$L_{simple,t} = E_{x_0 \sim q; z \sim \mathcal{N}(0, I)}\left[\left\|\epsilon_\theta(x_t, t) - z\right\|^2\right]$$

resulted in better models.

Backward diffusion process

After a noise prediction network is trained, it can be used for generating data points in the original distribution in a loop as follows:

1. Compute the noise estimate $\epsilon \leftarrow \epsilon_\theta(x_t, t)$
2. Compute the original data estimate $\tilde{x}_0 \leftarrow (x_t - \sigma_t \epsilon)/\sqrt{\bar{\alpha}_t}$
3. Sample the previous data $x_{t-1} \sim \mathcal{N}(\tilde{\mu}_t(x_t, \tilde{x}_0), \tilde{\sigma}_t^2 I)$
4. Change time $t \leftarrow t - 1$

Score-based generative model

Score-based generative model is another formulation of diffusion modelling. They are also called noise conditional score network (NCSN) or score-matching with Langevin dynamics (SMLD).[18][19][20][21]

Score matching

The idea of score functions

Consider the problem of image generation. Let x represent an image, and let $q(x)$ be the probability distribution over all possible images. If we have $q(x)$ itself, then we can say for certain how likely a certain image is. However, this is intractable in general.

Most often, we are uninterested in knowing the absolute probability of a certain image. Instead, we are usually only interested in knowing how likely a certain image is compared to its immediate neighbors — e.g. how much more likely is an image of cat compared to some small variants of it? Is it more likely if the image contains two whiskers, or three, or with some Gaussian noise added?

Consequently, we are actually quite uninterested in $q(x)$ itself, but rather, $\nabla_x \ln q(x)$. This has two major effects:

- One, we no longer need to normalize $q(x)$, but can use any $\tilde{q}(x) = Cq(x)$, where $C = \int \tilde{q}(x)dx > 0$ is any unknown constant that is of no concern to us.
- Two, we are comparing $q(x)$ neighbors $q(x+dx)$, by $\frac{q(x)}{q(x+dx)} = e^{-\langle \nabla_x \ln q, dx \rangle}$

Let the score function be $s(x) := \nabla_x \ln q(x)$; then consider what we can do with $s(x)$.

As it turns out, $s(x)$ allows us to sample from $q(x)$ using thermodynamics. Specifically, if we have a potential energy function $U(x) = -\ln q(x)$, and a lot of particles in the potential well, then the distribution at thermodynamic equilibrium is the Boltzmann distribution $q_U(x) \propto e^{-U(x)/k_BT} = q(x)^{1/k_BT}$. At temperature $k_BT = 1$, the Boltzmann distribution is exactly $q(x)$.

Therefore, to model $q(x)$, we may start with a particle sampled at any convenient distribution (such as the standard gaussian distribution), then simulate the motion of the particle forwards according to the Langevin equation

$$dx_t = -\nabla_{x_t} U(x_t)dt + dW_t$$

and the Boltzmann distribution is, by Fokker-Planck equation, the unique thermodynamic equilibrium. So no matter what distribution x_0 has, the distribution of x_t converges in distribution to q as $t \to \infty$.

Learning the score function

Given a density q, we wish to learn a score function approximation $f_\theta \approx \nabla \ln q$. This is **score matching**.[22] Typically, score matching is formalized as minimizing **Fisher divergence** function $E_q[\|f_\theta(x) - \nabla \ln q(x)\|^2]$. By expanding the integral, and performing an integration by parts,

$$E_q[\|f_\theta(x) - \nabla \ln q(x)\|^2] = E_q[\|f_\theta\|^2 + 2\nabla \cdot f_\theta] + C$$

giving us a loss function, also known as the Hyvärinen scoring rule, that can be minimized by stochastic gradient descent.

Annealing the score function

Suppose we need to model the distribution of images, and we want $x_0 \sim \mathcal{N}(0, I)$, a white-noise image. Now, most white-noise images do not look like real images, so $q(x_0) \approx 0$ for large swaths of $x_0 \sim \mathcal{N}(0, I)$. This presents a problem for learning the score function, because if there are no samples around a certain point, then we can't learn the score function at that point. If we do not know the score function $\nabla_{x_t} \ln q(x_t)$ at that point, then we cannot impose the time-evolution equation on a particle:

$$dx_t = \nabla_{x_t} \ln q(x_t) dt + dW_t$$

To deal with this problem, we perform annealing. If q is too different from a white-noise distribution, then progressively add noise until it is indistinguishable from one. That is, we perform a forward diffusion, then learn the score function, then use the score function to perform a backward diffusion.

Continuous diffusion processes

Forward diffusion process

Consider again the forward diffusion process, but this time in continuous time:

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

By taking the $\beta_t \to \beta(t)dt, \sqrt{dt} z_t \to dW_t$ limit, we obtain a continuous diffusion process, in the form of a stochastic differential equation:

$$dx_t = -\frac{1}{2}\beta(t) x_t dt + \sqrt{\beta(t)} dW_t$$

where W_t is a Wiener process (multidimensional Brownian motion).

Now, the equation is exactly a special case of the overdamped Langevin equation

$$dx_t = -\frac{D}{k_B T}(\nabla_x U) dt + \sqrt{2D} dW_t$$

where D is diffusion tensor, T is temperature, and U is potential energy field. If we substitute in $D = \frac{1}{2}\beta(t) I, k_B T = 1, U = \frac{1}{2}\|x\|^2$, we recover the above equation. This explains why the phrase "Langevin dynamics" is sometimes used in diffusion models.

Now the above equation is for the stochastic motion of a single particle. Suppose we have a cloud of particles distributed according to q at time $t = 0$, then after a long time, the cloud of particles would settle into the stable distribution of $\mathcal{N}(0, I)$. Let ρ_t be the density of the cloud of particles at time t, then we have

$$\rho_0 = q; \quad \rho_T \approx \mathcal{N}(0, I)$$

and the goal is to somehow reverse the process, so that we can start at the end and diffuse back to the beginning.

By Fokker-Planck equation, the density of the cloud evolves according to

$$\partial_t \ln \rho_t = \frac{1}{2}\beta(t) \left(n + (x + \nabla \ln \rho_t) \cdot \nabla \ln \rho_t + \Delta \ln \rho_t\right)$$

where n is the dimension of space, and Δ is the Laplace operator. Equivalently,

$$\partial_t \rho_t = \frac{1}{2}\beta(t)(\nabla \cdot (x\rho_t) + \Delta \rho_t)$$

Backward diffusion process

If we have solved ρ_t for time $t \in [0, T]$, then we can exactly reverse the evolution of the cloud. Suppose we start with another cloud of particles with density $\nu_0 = \rho_T$, and let the particles in the cloud evolve according to

$$dy_t = \frac{1}{2}\beta(T-t)y_t dt + \beta(T-t)\underbrace{\nabla_{y_t} \ln \rho_{T-t}(y_t)}_{\text{score function}} dt + \sqrt{\beta(T-t)} dW_t$$

then by plugging into the Fokker-Planck equation, we find that $\partial_t \rho_{T-t} = \partial_t \nu_t$. Thus this cloud of points is the original cloud, evolving backwards.[23]

Noise conditional score network (NCSN)

At the continuous limit,

$$\bar{\alpha}_t = (1-\beta_1)\cdots(1-\beta_t) = e^{\sum_i \ln(1-\beta_i)} \to e^{-\int_0^t \beta(t)dt}$$

and so

$$x_t | x_0 \sim N\left(e^{-\frac{1}{2}\int_0^t \beta(t)dt} x_0, \left(1 - e^{-\int_0^t \beta(t)dt}\right) I\right)$$

In particular, we see that we can directly sample from any point in the continuous diffusion process without going through the intermediate steps, by first sampling $x_0 \sim q, z \sim \mathcal{N}(0, I)$, then get $x_t = e^{-\frac{1}{2}\int_0^t \beta(t)dt} x_0 + \left(1 - e^{-\int_0^t \beta(t)dt}\right) z$. That is, we can quickly sample $x_t \sim \rho_t$ for any $t \geq 0$.

Now, define a certain probability distribution γ over $[0, \infty)$, then the score-matching loss function is defined as the expected Fisher divergence:

$$L(\theta) = E_{t\sim\gamma, x_t\sim\rho_t}[\|f_\theta(x_t, t)\|^2 + 2\nabla \cdot f_\theta(x_t, t)]$$

After training, $f_\theta(x_t, t) \approx \nabla \ln \rho_t$, so we can perform the backwards diffusion process by first sampling $x_T \sim \mathcal{N}(0, I)$, then integrating the SDE from $t = T$ to $t = 0$:

$$x_{t-dt} = x_t + \frac{1}{2}\beta(t)x_t dt + \beta(t) f_\theta(x_t, t)dt + \sqrt{\beta(t)} dW_t$$

This may be done by any SDE integration method, such as Euler–Maruyama method.

The name "noise conditional score network" is explained thus:

- "network", because f_θ is implemented as a neural network.
- "score", because the output of the network is interpreted as approximating the score function $\nabla \ln \rho_t$.
- "noise conditional", because ρ_t is equal to ρ_0 blurred by an added gaussian noise that increases with time, and so the score function depends on the amount of noise added.

Their equivalence

DDPM and score-based generative models are equivalent.[19][2][24] This means that a network trained using DDPM can be used as a NCSN, and vice versa.

We know that $x_t|x_0 \sim N\left(\sqrt{\bar{\alpha}_t}x_0, \sigma_t^2 I\right)$, so by Tweedie's formula, we have

$$\nabla_{x_t} \ln q(x_t) = \frac{1}{\sigma_t^2}(-x_t + \sqrt{\bar{\alpha}_t} E_q[x_0|x_t])$$

As described previously, the DDPM loss function is $\sum_t L_{simple,t}$ with

$$L_{simple,t} = E_{x_0 \sim q; z \sim \mathcal{N}(0,I)}\left[\|\epsilon_\theta(x_t,t) - z\|^2\right]$$

where $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sigma_t z$. By a change of variables,

$$L_{simple,t} = E_{x_0, x_t \sim q}\left[\left\|\epsilon_\theta(x_t,t) - \frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{\sigma_t}\right\|^2\right] = E_{x_t \sim q, x_0 \sim q(\cdot|x_t)}\left[\left\|\epsilon_\theta(x_t,t) - \frac{x_t - \sqrt{\bar{\alpha}_t}x_0}{\sigma_t}\right\|^2\right]$$

and the term inside becomes a least squares regression, so if the network actually reaches the global minimum of loss, then we have $\epsilon_\theta(x_t,t) = \frac{x_t - \sqrt{\bar{\alpha}_t}E_q[x_0|x_t]}{\sigma_t} = -\sigma_t \nabla_{x_t} \ln q(x_t)$

Thus, a score-based network predicts noise, and can be used for denoising.

Conversely, the continuous limit $x_{t-1} = x_{t-dt}, \beta_t = \beta(t)dt, z_t\sqrt{dt} = dW_t$ of the backward equation

$$x_{t-1} = \frac{x_t}{\sqrt{\alpha_t}} - \frac{\beta_t}{\sigma_t\sqrt{\alpha_t}}\epsilon_\theta(x_t,t) + \sqrt{\beta_t}z_t; \quad z_t \sim \mathcal{N}(0,I)$$

gives us precisely the same equation as score-based diffusion:

$$x_{t-dt} = x_t(1 + \beta(t)dt/2) + \beta(t)\nabla_{x_t} \ln q(x_t)dt + \sqrt{\beta(t)}dW_t$$

Thus, at infinitesimal steps of DDPM, a denoising network performs score-based diffusion.

Main variants

Noise schedule

In DDPM, the sequence of numbers $0 = \sigma_0 < \sigma_1 < \cdots < \sigma_T < 1$ is called a (discrete time) **noise schedule**. In general, consider a strictly increasing monotonic function σ of type $\mathbb{R} \to (0,1)$, such as the sigmoid function. In that case, a noise schedule is a sequence of real numbers $\lambda_1 < \lambda_2 < \cdots < \lambda_T$. It then defines a sequence of noises $\sigma_t := \sigma(\lambda_t)$, which then derives the other quantities $\beta_t = 1 - \frac{1-\sigma_t^2}{1-\sigma_{t-1}^2}$.

In order to use arbitrary noise schedules, instead of training a noise prediction model $\epsilon_\theta(x_t,t)$, one trains $\epsilon_\theta(x_t,\sigma_t)$.

Similarly, for the noise conditional score network, instead of training $f_\theta(x_t,t)$, one trains $f_\theta(x_t,\sigma_t)$.

Denoising Diffusion Implicit Model (DDIM)

The original DDPM method for generating images is slow, since the forward diffusion process usually takes $T \sim 1000$ to make the distribution of x_T to appear close to gaussian. However this means the backward diffusion process also take 1000 steps. Unlike the forward diffusion process, which can skip steps as $x_t|x_0$ is gaussian for all $t \geq 1$, the backward diffusion process does not allow skipping steps. For example, to sample $x_{t-2}|x_{t-1} \sim \mathcal{N}(\mu_\theta(x_{t-1}, t-1), \Sigma_\theta(x_{t-1}, t-1))$ requires the model to first sample x_{t-1}. Attempting to directly sample $x_{t-2}|x_t$ would require us to marginalize out x_{t-1}, which is generally intractable.

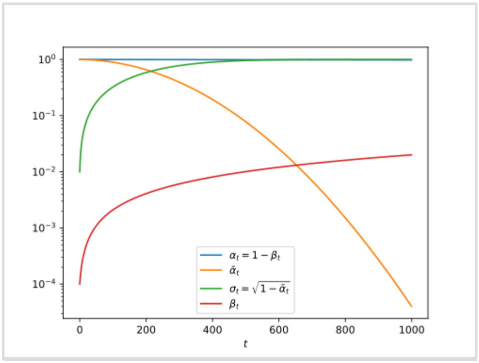


Illustration for a linear diffusion noise schedule. With settings $\beta_1 = 10^{-4}, \beta_{1000} = 0.02$.

DDIM[25] is a method to take any model trained on DDPM loss, and use it to sample with some steps skipped, sacrificing an adjustable amount of quality. If we generate the Markovian chain case in DDPM to non-Markovian case, DDIM corresponds to the case that the reverse process has variance equals to 0. In other words, the reverse process (and also the forward process) is deterministic. When using fewer sampling steps, DDIM outperforms DDPM.

In detail, the DDIM sampling method is as follows. Start with the forward diffusion process $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sigma_t\epsilon$. Then, during the backward denoising process, given $x_t, \epsilon_\theta(x_t, t)$, the original data is estimated as

$$x_0' = \frac{x_t - \sigma_t\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}$$

then the backward diffusion process can jump to any step $0 \leq s < t$, and the next denoised sample is

$$x_s = \sqrt{\bar{\alpha}_s}x_0' + \sqrt{\sigma_s^2 - (\sigma_s')^2}\epsilon_\theta(x_t, t) + \sigma_s'\epsilon$$

where σ_s' is an arbitrary real number within the range $[0, \sigma_s]$, and $\epsilon \sim \mathcal{N}(0, I)$ is a newly sampled gaussian noise.[17] If all $\sigma_s' = 0$, then the backward process becomes deterministic, and this special case of DDIM is also called "DDIM". The original paper noted that when the process is deterministic, samples generated with only 20 steps are already very similar to ones generated with 1000 steps on the high-level.

The original paper recommended defining a single "eta value" $\eta \in [0, 1]$, such that $\sigma_s' = \eta\tilde{\sigma}_s$. When $\eta = 1$, this is the original DDPM. When $\eta = 0$, this is the fully deterministic DDIM. For intermediate values, the process interpolates between them.

By the equivalence, the DDIM algorithm also applies for score-based diffusion models.

Latent diffusion model (LDM)

Since the diffusion model is a general method for modelling probability distributions, if one wants to model a distribution over images, one can first encode the images into a lower-dimensional space by an encoder, then use a diffusion model to model the distribution over encoded images. Then to generate an image, one can sample from the diffusion model, then use a decoder to decode it into an image.[26]

The encoder-decoder pair is most often a variational autoencoder (VAE).

Architectural improvements

[27] proposed various architectural improvements. For example, they proposed log-space interpolation during backward sampling. Instead of sampling from $x_{t-1} \sim \mathcal{N}(\tilde{\mu}_t(x_t, \tilde{x}_0), \tilde{\sigma}_t^2 I)$, they recommended sampling from $\mathcal{N}(\tilde{\mu}_t(x_t, \tilde{x}_0), (\sigma_t^v \tilde{\sigma}_t^{1-v})^2 I)$ for a learned parameter v.

In the *v-prediction* formalism, the noising formula $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t$ is reparameterised by an angle ϕ_t such that $\cos \phi_t = \sqrt{\bar{\alpha}_t}$ and a "velocity" defined by $\cos \phi_t \epsilon_t - \sin \phi_t x_0$. The network is trained to predict the velocity $\hat{v}_\theta$, and denoising is by $x_{\phi_t - \delta} = \cos(\delta)\, x_{\phi_t} - \sin(\delta) \hat{v}_\theta\, (x_{\phi_t})$.[28] This parameterization was found to improve performance, as the model can be trained to reach total noise (i.e. $\phi_t = 90^\circ$) and then reverse it, whereas the standard parameterization never reaches total noise since $\sqrt{\bar{\alpha}_t} > 0$ is always true.[29]

Classifier guidance

Classifier guidance was proposed in 2021 to improve class-conditional generation by using a classifier. The original publication used CLIP text encoders to improve text-conditional image generation.[30]

Suppose we wish to sample not from the entire distribution of images, but conditional on the image description. We don't want to sample a generic image, but an image that fits the description "black cat with red eyes". Generally, we want to sample from the distribution $p(x|y)$, where x ranges over images, and y ranges over classes of images (a description "black cat with red eyes" is just a very detailed class, and a class "cat" is just a very vague description).

Taking the perspective of the noisy channel model, we can understand the process as follows: To generate an image x conditional on description y, we imagine that the requester really had in mind an image x, but the image is passed through a noisy channel and came out garbled, as y. Image generation is then nothing but inferring which x the requester had in mind.

In other words, conditional image generation is simply "translating from a textual language into a pictorial language". Then, as in noisy-channel model, we use Bayes theorem to get

$$p(x|y) \propto p(y|x) p(x)$$

in other words, if we have a good model of the space of all images, and a good image-to-class translator, we get a class-to-image translator "for free". In the equation for backward diffusion, the score $\nabla \ln p(x)$ can be replaced by

$$\nabla_x \ln p(x|y) = \underbrace{\nabla_x \ln p(x)}_{\text{score}} + \underbrace{\nabla_x \ln p(y|x)}_{\text{classifier guidance}}$$

where $\nabla_x \ln p(x)$ is the score function, trained as previously described, and $\nabla_x \ln p(y|x)$ is found by using a differentiable image classifier.

During the diffusion process, we need to condition on the time, giving

$$\nabla_{x_t} \ln p(x_t|y, t) = \nabla_{x_t} \ln p(y|x_t, t) + \nabla_{x_t} \ln p(x_t|t)$$

Although, usually the classifier model does not depend on time, in which case $p(y|x_t, t) = p(y|x_t)$.

Classifier guidance is defined for the gradient of score function, thus for score-based diffusion network, but as previously noted, score-based diffusion models are equivalent to denoising models by $\epsilon_\theta(x_t, t) = -\sigma_t \nabla_{x_t} \ln p(x_t|t)$, and similarly, $\epsilon_\theta(x_t, y, t) = -\sigma_t \nabla_{x_t} \ln p(x_t|y, t)$. Therefore, classifier guidance works for denoising diffusion as well, using the modified noise prediction:[30]

$$\epsilon_\theta(x_t, y, t) = \epsilon_\theta(x_t, t) - \underbrace{\sigma_t \nabla_{x_t} \ln p(y|x_t, t)}_{\text{classifier guidance}}$$

With temperature

The classifier-guided diffusion model samples from $p(x|y)$, which is concentrated around the maximum a posteriori estimate $\arg\max_x p(x|y)$. If we want to force the model to move towards the maximum likelihood estimate $\arg\max_x p(y|x)$, we can use

$$p_\gamma(x|y) \propto p(y|x)^\gamma p(x)$$

where $\gamma > 0$ is interpretable as *inverse temperature*. In the context of diffusion models, it is usually called the **guidance scale**. A high γ would force the model to sample from a distribution concentrated around $\arg\max_x p(y|x)$. This sometimes improves quality of generated images.[30]

This gives a modification to the previous equation:

$$\nabla_x \ln p_\beta(x|y) = \nabla_x \ln p(x) + \gamma \nabla_x \ln p(y|x)$$

For denoising models, it corresponds to[31]

$$\epsilon_\theta(x_t, y, t) = \epsilon_\theta(x_t, t) - \gamma \sigma_t \nabla_{x_t} \ln p(y|x_t, t)$$

Classifier-free guidance (CFG)

If we do not have a classifier $p(y|x)$, we could still extract one out of the image model itself:[31]

$$\nabla_x \ln p_\gamma(x|y) = (1 - \gamma) \nabla_x \ln p(x) + \gamma \nabla_x \ln p(x|y)$$

Such a model is usually trained by presenting it with both (x, y) and (x, None), allowing it to model both $\nabla_x \ln p(x|y)$ and $\nabla_x \ln p(x)$.

Note that for CFG, the diffusion model cannot be merely a generative model of the entire data distribution $\nabla_x \ln p(x)$. It must be a conditional generative model $\nabla_x \ln p(x|y)$. For example, in stable diffusion, the diffusion backbone takes as input both a noisy model x_t, a time t, and a conditioning vector y (such as a vector encoding a text prompt), and produces a noise prediction $\epsilon_\theta(x_t, y, t)$.

For denoising models, it corresponds to

$$\epsilon_\theta(x_t, y, t, \gamma) = \epsilon_\theta(x_t, t) + \gamma(\epsilon_\theta(x_t, y, t) - \epsilon_\theta(x_t, t))$$

As sampled by DDIM, the algorithm can be written as[32]

$$\begin{aligned} \epsilon_{\text{uncond}} &\leftarrow \epsilon_\theta(x_t, t) \\ \epsilon_{\text{cond}} &\leftarrow \epsilon_\theta(x_t, t, c) \\ \epsilon_{\text{CFG}} &\leftarrow \epsilon_{\text{uncond}} + \gamma(\epsilon_{\text{cond}} - \epsilon_{\text{uncond}}) \\ x_0 &\leftarrow (x_t - \sigma_t \epsilon_{\text{CFG}}) / \sqrt{1 - \sigma_t^2} \end{aligned}$$

$$x_s \leftarrow \sqrt{1-\sigma_s^2}x_0 + \sqrt{\sigma_s^2 - (\sigma_s')^2}\epsilon_{\text{uncond}} + \sigma_s'\epsilon$$

A similar technique applies to language model sampling. Also, if the unconditional generation $\epsilon_{\text{uncond}} \leftarrow \epsilon_\theta(x_t, t)$ is replaced by $\epsilon_{\text{neg cond}} \leftarrow \epsilon_\theta(x_t, t, c')$, then it results in negative prompting, which pushes the generation away from c' condition.[33][34]

Samplers

Given a diffusion model, one may regard it either as a continuous process, and sample from it by integrating a SDE, or one can regard it as a discrete process, and sample from it by iterating the discrete steps. The choice of the "**noise schedule**" β_t can also affect the quality of samples. A noise schedule is a function that sends a natural number to a noise level:

$$t \mapsto \beta_t, \quad t \in \{1, 2, \dots\}, \beta \in (0,1)$$

A noise schedule is more often specified by a map $t \mapsto \sigma_t$. The two definitions are equivalent, since $\beta_t = 1 - \frac{1-\sigma_t^2}{1-\sigma_{t-1}^2}$.

In the DDPM perspective, one can use the DDPM itself (with noise), or DDIM (with adjustable amount of noise). The case where one adds noise is sometimes called ancestral sampling.[35] One can interpolate between noise and no noise. The amount of noise is denoted η ("eta value") in the DDIM paper, with $\eta = 0$ denoting no noise (as in *deterministic* DDIM), and $\eta = 1$ denoting full noise (as in DDPM).

In the perspective of SDE, one can use any of the numerical integration methods, such as Euler–Maruyama method, Heun's method, linear multistep methods, etc. Just as in the discrete case, one can add an adjustable amount of noise during the integration.[36]

A survey and comparison of samplers in the context of image generation is in.[37]

Other examples

Notable variants include[38] Poisson flow generative model,[39] consistency model,[40] critically-damped Langevin diffusion,[41] GenPhys,[42] cold diffusion,[43] discrete diffusion,[44][45] etc.

Flow-based diffusion model

Abstractly speaking, the idea of diffusion model is to take an unknown probability distribution (the distribution of natural-looking images), then progressively convert it to a known probability distribution (standard gaussian distribution), by building an absolutely continuous probability path connecting them. The probability path is in fact defined implicitly by the score function $\nabla \ln p_t$.

In denoising diffusion models, the forward process adds noise, and the backward process removes noise. Both the forward and backward processes are SDEs, though the forward process is integrable in closed-form, so it can be done at no computational cost. The backward process is not integrable in closed-form, so it must be integrated step-by-step by standard SDE solvers, which can be very expensive. The probability path in diffusions model is defined through an Itô process and one can retrieve the deterministic process by using the Probability ODE flow formulation.[2]

In flow-based diffusion models, the forward process is a deterministic flow along a time-dependent vector field, and the backward process is also a deterministic flow along the same vector field, but going backwards. Both processes are solutions to ODEs. If the vector field is well-behaved, the ODE will also be well-behaved.

Given two distributions π_0 and π_1, a flow-based model is a time-dependent velocity field $v_t(x)$ in $[0,1] \times \mathbb{R}^d$, such that if we start by sampling a point $x \sim \pi_0$, and let it move according to the velocity field:

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)) \quad t \in [0,1], \quad \text{starting from } \phi_0(x) = x$$

we end up with a point $x_1 \sim \pi_1$. The solution ϕ_t of the above ODE define a probability path $p_t = [\phi_t]_{\#}\pi_0$ by the pushforward measure operator. In particular, $[\phi_1]_{\#}\pi_0 = \pi_1$.

The probability path and the velocity field also satisfy the continuity equation, in the sense of probability distribution:

$$\partial_t p_t + \nabla \cdot (v_t p_t) = 0$$

To construct a probability path, we start by construct a conditional probability path $p_t(x|z)$ and the corresponding conditional velocity field $v_t(x|z)$ on some conditional distribution $q(z)$. A natural choice is the Gaussian conditional probability path:

$$p_t(x|z) = \mathcal{N}\left(m_t(z), \zeta_t^2 I\right)$$

The conditional velocity field which corresponds to the geodesic path between conditional Gaussian path is

$$v_t(x|z) = \frac{\zeta_t'}{\zeta_t}(x - m_t(z)) + m_t'(z)$$

The probability path and velocity field are then computed by marginalizing

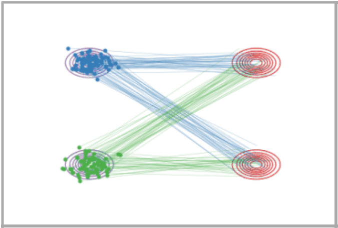
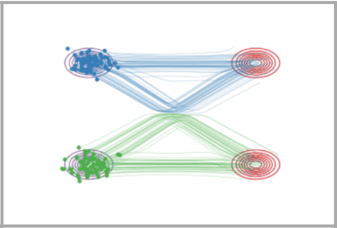
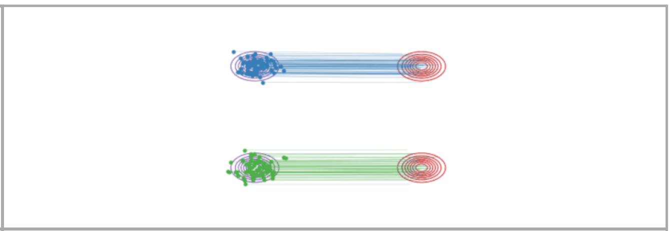
$$p_t(x) = \int p_t(x|z)q(z)dz \qquad \text{and} \qquad v_t(x) = \mathbb{E}_{q(z)}\left[\frac{v_t(x|z)p_t(x|z)}{p_t(x)}\right]$$

Optimal transport flow

The idea of **optimal transport flow** [46] is to construct a probability path minimizing the Wasserstein metric. The distribution on which we condition is an approximation of the optimal transport plan between π_0 and π_1: $z = (x_0, x_1)$ and $q(z) = \Gamma(\pi_0, \pi_1)$, where Γ is the optimal transport plan, which can be approximated by **mini-batch optimal transport.** If the batch size is not large, then the transport it computes can be very far from the true optimal transport.

Rectified flow

The idea of **rectified flow**[47][48] is to learn a flow model such that the velocity is nearly constant along each flow path. This is beneficial, because we can integrate along such a vector field with very few steps. For example, if an ODE $\dot{\phi}_t(x) = v_t(\phi_t(x))$ follows perfectly straight paths, it simplifies to $\phi_t(x) = x_0 + t \cdot v_0(x_0)$, allowing for exact solutions in one step. In practice, we cannot reach such perfection, but when the flow field is nearly so, we can take a few large steps instead of many little steps.

		
Linear interpolation	Rectified Flow	Straightened Rectified Flow [1] (https://www.cs.utexas.edu/~lqiang/rectflow/html/intro.html)

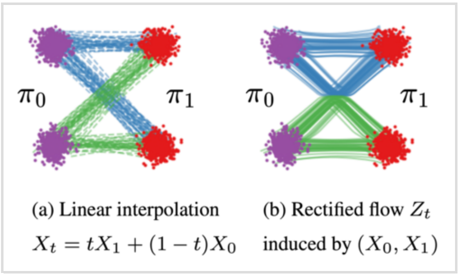
The general idea is to start with two distributions π_0 and π_1, then construct a flow field $\phi^0 = \{\phi_t : t \in [0,1]\}$ from it, then repeatedly apply a "reflow" operation to obtain successive flow fields $\phi^1, \phi^2, \dots$, each straighter than the previous one. When the flow field is straight enough for the application, we stop.

Generally, for any time-differentiable process ϕ_t, v_t can be estimated by solving:

$$\min_\theta \int_0^1 \mathbb{E}_{x \sim p_t} \left[\| v_t(x,\theta) - v_t(x) \|^2 \right] \, \mathrm{d}t.$$

In rectified flow, by injecting strong priors that intermediate trajectories are straight, it can achieve both theoretical relevance for optimal transport and computational efficiency, as ODEs with straight paths can be simulated precisely without time discretization.

Specifically, rectified flow seeks to match an ODE with the marginal distributions of the **linear interpolation** between points from distributions π_0 and π_1. Given observations $x_0 \sim \pi_0$ and $x_1 \sim \pi_1$, the canonical linear interpolation $x_t = tx_1 + (1-t)x_0, t \in [0,1]$ yields a trivial case $\dot{x}_t = x_1 - x_0$, which cannot be causally simulated without x_1. To address this, x_t is "projected" into a space of causally simulatable ODEs, by minimizing the least squares loss with respect to the direction $x_1 - x_0$:


π_0
π_1
π_0
π_1
(a) Linear interpolation $X_t = tX_1 + (1-t)X_0$
(b) Rectified flow Z_t induced by (X_0, X_1)

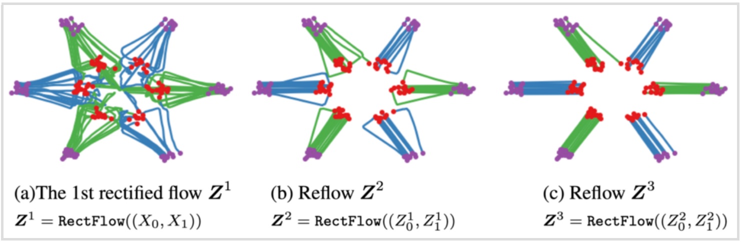
Transport by rectified flow[47]

$$\min_\theta \int_0^1 \mathbb{E}_{\pi_0, \pi_1, p_t} \left[\| (x_1 - x_0) - v_t(x_t) \|^2 \right] \, \mathrm{d}t.$$

The data pair (x_0, x_1) can be any coupling of π_0 and π_1, typically independent (i.e., $(x_0, x_1) \sim \pi_0 \times \pi_1$) obtained by randomly combining observations from π_0 and π_1. This process ensures that the trajectories closely mirror the density map of x_t trajectories but *reroute* at intersections to ensure causality. This rectifying process is also known as Flow Matching,[49] Stochastic Interpolation,[50] and alpha-(de)blending.[51]

A distinctive aspect of rectified flow is its capability for "**reflow**", which straightens the trajectory of ODE paths. Denote the rectified flow $\phi^0 = \{\phi_t : t \in [0,1]\}$ induced from (x_0, x_1) as $\phi^0 = \mathsf{Rectflow}((x_0, x_1))$. Recursively applying this $\mathsf{Rectflow}(\cdot)$ operator generates a series of rectified flows $\phi^{k+1} = \mathsf{Rectflow}((\phi_0^k(x_0), \phi_1^k(x_1)))$. This "reflow" process not only reduces transport costs but also straightens the paths of rectified flows, making ϕ^k paths straighter with increasing k.

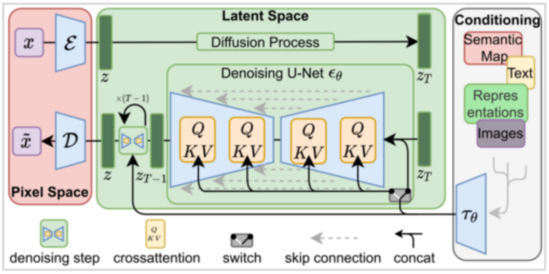
Rectified flow includes a nonlinear extension where linear interpolation x_t is replaced with any time-differentiable curve that connects x_0 and x_1, given by $x_t = \alpha_t x_1 + \beta_t x_0$. This framework encompasses DDIM and probability flow ODEs as special cases, with particular choices of α_t and β_t. However, in the case where the path of x_t is not straight, the reflow process no longer ensures a reduction in convex transport costs, and also no longer straighten the paths of ϕ_t.[47]


(a)The 1st rectified flow Z^1
$Z^1 = \texttt{RectFlow}((X_0, X_1))$
(b) Reflow Z^2
$Z^2 = \texttt{RectFlow}((Z_0^1, Z_1^1))$
(c) Reflow Z^3
$Z^3 = \texttt{RectFlow}((Z_0^2, Z_1^2))$
The reflow process[47]

See [52] for a tutorial on flow matching, with animations.

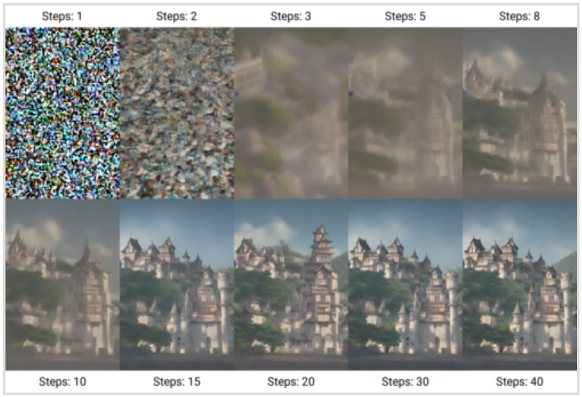
Choice of architecture

Diffusion model



Architecture of Stable Diffusion

For generating images by DDPM, we need a neural network that takes a time t and a noisy image x_t, and predicts a noise $\epsilon_\theta(x_t, t)$ from it. Since predicting the noise is the same as predicting the denoised image, then subtracting it from x_t, denoising architectures tend to work well. For example, the U-Net, which was found to be good for denoising images, is often used for denoising diffusion models that generate images.[53]



The denoising process used by Stable Diffusion

For DDPM, the underlying architecture ("backbone") does not have to be a U-Net. It just has to predict the noise somehow. For example, the diffusion transformer (DiT) uses a Transformer to predict the mean and diagonal covariance of the noise, given the textual conditioning and the partially denoised image. It is the same as standard U-Net-based denoising diffusion model, with a Transformer replacing the U-Net.[54] Mixture of experts-Transformer can also be applied.[55]

DDPM can be used to model general data distributions, not just natural-looking images. For example, Human Motion Diffusion[56] models human motion trajectory by DDPM. Each human motion trajectory is a sequence of poses, represented by either joint rotations or positions. It uses a Transformer network to generate a less noisy trajectory out of a noisy one.

Conditioning

The base diffusion model can only generate unconditionally from the whole distribution. For example, a diffusion model learned on ImageNet would generate images that look like a random image from ImageNet. To generate images from just one category, one would need to impose the condition, and then sample from the conditional distribution. Whatever condition one wants to impose, one needs to first convert the conditioning into a vector of floating point numbers, then feed it into the underlying diffusion model neural network. However, one has freedom in choosing how to convert the conditioning into a vector.

Stable Diffusion, for example, imposes conditioning in the form of cross-attention mechanism, where the query is an intermediate representation of the image in the U-Net, and both key and value are the conditioning vectors. The conditioning can be selectively applied to only parts of an image, and new kinds of conditionings can be finetuned upon the base model, as used in ControlNet.[57]

As a particularly simple example, consider image inpainting. The conditions are $\tilde{x}$, the reference image, and m, the inpainting mask. The conditioning is imposed at each step of the backward diffusion process, by first sampling $\tilde{x}_t \sim N\left(\sqrt{\bar{\alpha}_t}\tilde{x}, \sigma_t^2 I\right)$, a noisy version of $\tilde{x}$, then replacing x_t with $(1-m) \odot x_t + m \odot \tilde{x}_t$, where $\odot$ means elementwise multiplication.[58] Another application of cross-attention mechanism is prompt-to-prompt image editing.[59]

Conditioning is not limited to just generating images from a specific category, or according to a specific caption (as in text-to-image). For example,[56] demonstrated generating human motion, conditioned on an audio clip of human walking (allowing syncing motion to a soundtrack), or video of human running, or a text description of human motion, etc. For how conditional diffusion models are mathematically formulated, see a methodological summary in.[60]

Upscaling

As generating an image takes a long time, one can try to generate a small image by a base diffusion model, then upscale it by other models. Upscaling can be done by GAN,[61] Transformer,[62] or signal processing methods like Lanczos resampling.

Diffusion models themselves can be used to perform upscaling. Cascading diffusion model stacks multiple diffusion models one after another, in the style of Progressive GAN. The lowest level is a standard diffusion model that generate 32x32 image, then the image would be upscaled by a diffusion model specifically trained for upscaling, and the process repeats.[53]

In more detail, the diffusion upscaler is trained as follows:[53]

- Sample (x_0, z_0, c), where x_0 is the high-resolution image, z_0 is the same image but scaled down to a low-resolution, and c is the conditioning, which can be the caption of the image, the class of the image, etc.
- Sample two white noises ϵ_x, ϵ_z, two time-steps t_x, t_z. Compute the noisy versions of the high-resolution and low-resolution images: $\begin{cases} x_{t_x} &= \sqrt{\bar{\alpha}_{t_x}} x_0 + \sigma_{t_x}\epsilon_x \\ z_{t_z} &= \sqrt{\bar{\alpha}_{t_z}} z_0 + \sigma_{t_z}\epsilon_z \end{cases}$.
- Train the denoising network to predict ϵ_x given $x_{t_x}, z_{t_z}, t_x, t_z, c$. That is, apply gradient descent on θ on the L2 loss $\|\epsilon_\theta(x_{t_x}, z_{t_z}, t_x, t_z, c) - \epsilon_x\|_2^2$.

Examples

This section collects some notable diffusion models, and briefly describes their architecture.

OpenAI

The DALL-E series by OpenAI are text-conditional diffusion models of images.

The first version of DALL-E (2021) is not actually a diffusion model. Instead, it uses a Transformer architecture that autoregressively generates a sequence of tokens, which is then converted to an image by the decoder of a discrete VAE. Released with DALL-E was the CLIP classifier, which was used by DALL-E to rank generated images according to how close the image fits the text.

GLIDE (2022-03)[63] is a 3.5-billion diffusion model, and a small version was released publicly.[6] Soon after, DALL-E 2 was released (2022-04).[64] DALL-E 2 is a 3.5-billion cascaded diffusion model that generates images from text by "inverting the CLIP image encoder", the technique which they termed "unCLIP".

The unCLIP method contains 4 models: a CLIP image encoder, a CLIP text encoder, an image decoder, and a "prior" model (which can be a diffusion model, or an autoregressive model). During training, the prior model is trained to convert CLIP image encodings to CLIP text encodings. The image decoder is trained to convert CLIP image encodings back to images. During inference, a text is converted by the CLIP text encoder to a vector, then it is converted by the prior model to an image encoding, then it is converted by the image decoder to an image.

Sora (2024-02) is a diffusion Transformer model (DiT).

Stability AI

Stable Diffusion (2022-08), released by Stability AI, consists of a denoising latent diffusion model (860 million parameters), a VAE, and a text encoder. The denoising network is a U-Net, with cross-attention blocks to allow for conditional image generation.[65][26]

Stable Diffusion 3 (2024-03)[66] changed the latent diffusion model from the UNet to a Transformer model, and so it is a DiT. It uses rectified flow.

Stable Video 4D (2024-07)[67] is a latent diffusion model for videos of 3D objects.

Google

Imagen (2022)[68][69] uses a T5-XXL language model to encode the input text into an embedding vector. It is a cascaded diffusion model with three sub-models. The first step denoises a white noise to a 64×64 image, conditional on the embedding vector of the text. This model has 2B parameters. The second step upscales the image by 64×64→256×256, conditional on embedding. This model has 650M parameters. The third step is similar, upscaling by 256×256→1024×1024. This model has 400M parameters. The three denoising networks are all U-Nets.

Muse (2023-01)[70] is not a diffusion model, but an encoder-only Transformer that is trained to predict masked image tokens from unmasked image tokens.

Imagen 2 (2023-12) is also diffusion-based. It can generate images based on a prompt that mixes images and text. No further information available.[71] Imagen 3 (2024-05) is too. No further information available.[72]

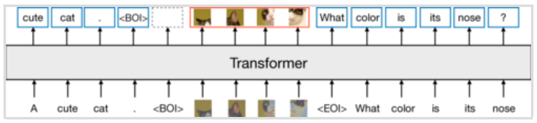
Veo (2024) generates videos by latent diffusion. The diffusion is conditioned on a vector that encodes both a text prompt and an image prompt.[73]

Meta

Make-A-Video (2022) is a text-to-video diffusion model.[74][75]

CM3leon (2023) is not a diffusion model, but an autoregressive causally masked Transformer, with mostly the same architecture as LLaMa-2.[76][77]

Transfusion (2024) is a Transformer that combines autoregressive text generation and denoising diffusion. Specifically, it generates text autoregressively (with causal masking), and generates images by denoising multiple times over image tokens (with all-to-all attention).[78]



Transfusion architectural diagram

Movie Gen (2024) is a series of Diffusion Transformers operating on latent space and by flow matching.[79]

See also

- Diffusion process
- Markov chain
- Variational inference
- Variational autoencoder

Further reading

- Review papers
  - Yang, Ling (2024-09-06), *YangLing0818/Diffusion-Models-Papers-Survey-Taxonomy* (https://github.com/YangLing0818/Diffusion-Models-Papers-Survey-Taxonomy), retrieved 2024-09-06
  - Yang, Ling; Zhang, Zhilong; Song, Yang; Hong, Shenda; Xu, Runsheng; Zhao, Yue; Zhang, Wentao; Cui, Bin; Yang, Ming-Hsuan (2023-11-09). "Diffusion Models: A Comprehensive Survey of Methods and Applications" (https://dl.acm.org/doi/abs/10.1145/3626235). *ACM Comput. Surv*. **56** (4): 105:1–105:39. arXiv:2209.00796 (https://arxiv.org/abs/2209.00796). doi:10.1145/3626235 (https://doi.org/10.1145%2F3626235). ISSN 0360-0300 (https://search.worldcat.org/issn/0360-0300).
  - Austin, Jacob; Johnson, Daniel D.; Ho, Jonathan; Tarlow, Daniel; Rianne van den Berg (2021). "Structured Denoising Diffusion Models in Discrete State-Spaces". arXiv:2107.03006 (https://arxiv.org/abs/2107.03006) [cs.LG (https://arxiv.org/archive/cs.LG)].
  - Croitoru, Florinel-Alin; Hondru, Vlad; Ionescu, Radu Tudor; Shah, Mubarak (2023-09-01). "Diffusion Models in Vision: A Survey" (https://ieeexplore.ieee.org/document/10081412). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **45** (9): 10850–10869. arXiv:2209.04747 (https://arxiv.org/abs/2209.04747). doi:10.1109/TPAMI.2023.3261988 (https://doi.org/10.1109%2FTPAMI.2023.3261988). ISSN 0162-8828 (https://search.worldcat.org/issn/0162-8828). PMID 37030794 (https://pubmed.ncbi.nlm.nih.gov/37030794).
- Mathematical details omitted in the article.

  - "Power of Diffusion Models" (https://astralord.github.io/posts/power-of-diffusion-models/). *AstraBlog*. 2022-09-25. Retrieved 2023-09-25.
  - Luo, Calvin (2022-08-25). "Understanding Diffusion Models: A Unified Perspective". arXiv:2208.11970 (https://arxiv.org/abs/2208.11970) [cs.LG (https://arxiv.org/archive/cs.LG)].
  - Weng, Lilian (2021-07-11). "What are Diffusion Models?" (https://lilianweng.github.io/posts/2021-07-11-diffusion-models/). *lilianweng.github.io*. Retrieved 2023-09-25.
- Tutorials
  - Nakkiran, Preetum; Bradley, Arwen; Zhou, Hattie; Advani, Madhu (2024). "Step-by-Step Diffusion: An Elementary Tutorial". arXiv:2406.08929 (https://arxiv.org/abs/2406.08929) [cs.LG (https://arxiv.org/archive/cs.LG)].
  - "Guidance: a cheat code for diffusion models" (https://benanne.github.io/2022/05/26/guidance.html). 26 May 2022. Overview of classifier guidance and classifier-free guidance, light on mathematical details.

References

1. Chang, Ziyi; Koulieris, George Alex; Shum, Hubert P. H. (2023). "On the Design Fundamentals of Diffusion Models: A Survey". arXiv:2306.04542 (https://arxiv.org/abs/2306.04542) [cs.LG (https://arxiv.org/archive/cs.LG)].
2. Song, Yang; Sohl-Dickstein, Jascha; Kingma, Diederik P.; Kumar, Abhishek; Ermon, Stefano; Poole, Ben (2021-02-10). "Score-Based Generative Modeling through Stochastic Differential Equations". arXiv:2011.13456 (https://arxiv.org/abs/2011.13456) [cs.LG (https://arxiv.org/archive/cs.LG)].
3. Croitoru, Florinel-Alin; Hondru, Vlad; Ionescu, Radu Tudor; Shah, Mubarak (2023). "Diffusion Models in Vision: A Survey". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **45** (9): 10850–10869. arXiv:2209.04747 (https://arxiv.org/abs/2209.04747). doi:10.1109/TPAMI.2023.3261988 (https://doi.org/10.1109%2FTPAMI.2023.3261988). PMID 37030794 (https://pubmed.ncbi.nlm.nih.gov/37030794). S2CID 252199918 (https://api.semanticscholar.org/CorpusID:252199918).
4. Ho, Jonathan; Jain, Ajay; Abbeel, Pieter (2020). "Denoising Diffusion Probabilistic Models" (https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html). *Advances in Neural Information Processing Systems*. **33**. Curran Associates, Inc.: 6840–6851.
5. Gu, Shuyang; Chen, Dong; Bao, Jianmin; Wen, Fang; Zhang, Bo; Chen, Dongdong; Yuan, Lu; Guo, Baining (2021). "Vector Quantized Diffusion Model for Text-to-Image Synthesis". arXiv:2111.14822 (https://arxiv.org/abs/2111.14822) [cs.CV (https://arxiv.org/archive/cs.CV)].
6. *GLIDE* (https://github.com/openai/glide-text2im), OpenAI, 2023-09-22, retrieved 2023-09-24
7. Nie, Shen; Zhu, Fengqi; Du, Chao; Pang, Tianyu; Liu, Qian; Zeng, Guangtao; Lin, Min; Li, Chongxuan (2024). "Scaling up Masked Diffusion Models on Text". arXiv:2410.18514 (https://arxiv.org/abs/2410.18514) [cs.AI (https://arxiv.org/archive/cs.AI)].
8. Li, Yifan; Zhou, Kun; Zhao, Wayne Xin; Wen, Ji-Rong (August 2023). "Diffusion Models for Non-autoregressive Text Generation: A Survey" (https://dx.doi.org/10.24963/ijcai.2023/750). *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. California: International Joint Conferences on Artificial Intelligence Organization. pp. 6692–6701. arXiv:2303.06574 (https://arxiv.org/abs/2303.06574). doi:10.24963/ijcai.2023/750 (https://doi.org/10.24963%2Fijcai.2023%2F750). ISBN 978-1-956792-03-4.
9. Han, Xiaochuang; Kumar, Sachin; Tsvetkov, Yulia (2023). "SSD-LM: Semi-autoregressive Simplex-based Diffusion Language Model for Text Generation and Modular Control" (https://dx.doi.org/10.18653/v1/2023.acl-long.647). *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Stroudsburg, PA, USA: Association for Computational Linguistics: 11575–11596. arXiv:2210.17432 (https://arxiv.org/abs/2210.17432). doi:10.18653/v1/2023.acl-long.647 (https://doi.org/10.18653%2Fv1%2F2023.acl-long.647).

10. Xu, Weijie; Hu, Wenxiang; Wu, Fanyou; Sengamedu, Srinivasan (2023). "DeTiME: Diffusion-Enhanced Topic Modeling using Encoder-decoder based LLM" (https://dx.doi.org/10.18653/v1/2023.findings-emnlp.606). *Findings of the Association for Computational Linguistics: EMNLP 2023*. Stroudsburg, PA, USA: Association for Computational Linguistics: 9040–9057. arXiv:2310.15296 (https://arxiv.org/abs/2310.15296). doi:10.18653/v1/2023.findings-emnlp.606 (https://doi.org/10.18653%2Fv1%2F2023.findings-emnlp.606).
11. Zhang, Haopeng; Liu, Xiao; Zhang, Jiawei (2023). "DiffuSum: Generation Enhanced Extractive Summarization with Diffusion" (https://dx.doi.org/10.18653/v1/2023.findings-acl.828). *Findings of the Association for Computational Linguistics: ACL 2023*. Stroudsburg, PA, USA: Association for Computational Linguistics: 13089–13100. arXiv:2305.01735 (https://arxiv.org/abs/2305.01735). doi:10.18653/v1/2023.findings-acl.828 (https://doi.org/10.18653%2Fv1%2F2023.findings-acl.828).
12. Yang, Dongchao; Yu, Jianwei; Wang, Helin; Wang, Wen; Weng, Chao; Zou, Yuexian; Yu, Dong (2023). "Diffsound: Discrete Diffusion Model for Text-to-Sound Generation" (https://dx.doi.org/10.1109/taslp.2023.3268730). *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. **31**: 1720–1733. arXiv:2207.09983 (https://arxiv.org/abs/2207.09983). doi:10.1109/taslp.2023.3268730 (https://doi.org/10.1109%2Ftaslp.2023.3268730). ISSN 2329-9290 (https://search.worldcat.org/issn/2329-9290).
13. Janner, Michael; Du, Yilun; Tenenbaum, Joshua B.; Levine, Sergey (2022-12-20). "Planning with Diffusion for Flexible Behavior Synthesis". arXiv:2205.09991 (https://arxiv.org/abs/2205.09991) [cs.LG (https://arxiv.org/archive/cs.LG)].
14. Chi, Cheng; Xu, Zhenjia; Feng, Siyuan; Cousineau, Eric; Du, Yilun; Burchfiel, Benjamin; Tedrake, Russ; Song, Shuran (2024-03-14). "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". arXiv:2303.04137 (https://arxiv.org/abs/2303.04137) [cs.RO (https://arxiv.org/archive/cs.RO)].
15. Sohl-Dickstein, Jascha; Weiss, Eric; Maheswaranathan, Niru; Ganguli, Surya (2015-06-01). "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" (http://proceedings.mlr.press/v37/sohl-dickstein15.pdf) (PDF). *Proceedings of the 32nd International Conference on Machine Learning*. **37**. PMLR: 2256–2265. arXiv:1503.03585 (https://arxiv.org/abs/1503.03585).
16. Ho, Jonathan (Jun 20, 2020), *hojonathanho/diffusion* (https://github.com/hojonathanho/diffusion), retrieved 2024-09-07
17. Weng, Lilian (2021-07-11). "What are Diffusion Models?" (https://lilianweng.github.io/posts/2021-07-11-diffusion-models/). *lilianweng.github.io*. Retrieved 2023-09-24.
18. "Generative Modeling by Estimating Gradients of the Data Distribution | Yang Song" (https://yang-song.net/blog/2021/score/). *yang-song.net*. Retrieved 2023-09-24.
19. Song, Yang; Ermon, Stefano (2019). "Generative Modeling by Estimating Gradients of the Data Distribution" (https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html). *Advances in Neural Information Processing Systems*. **32**. Curran Associates, Inc. arXiv:1907.05600 (https://arxiv.org/abs/1907.05600).
20. Song, Yang; Sohl-Dickstein, Jascha; Kingma, Diederik P.; Kumar, Abhishek; Ermon, Stefano; Poole, Ben (2021-02-10). "Score-Based Generative Modeling through Stochastic Differential Equations". arXiv:2011.13456 (https://arxiv.org/abs/2011.13456) [cs.LG (https://arxiv.org/archive/cs.LG)].
21. *ermongroup/ncsn* (https://github.com/ermongroup/ncsn), ermongroup, 2019, retrieved 2024-09-07
22. "Sliced Score Matching: A Scalable Approach to Density and Score Estimation | Yang Song" (https://yang-song.net/blog/2019/ssm/). *yang-song.net*. Retrieved 2023-09-24.
23. Anderson, Brian D.O. (May 1982). "Reverse-time diffusion equation models" (https://dx.doi.org/10.1016/0304-4149(82)90051-5). *Stochastic Processes and Their Applications*. **12** (3): 313–326. doi:10.1016/0304-4149(82)90051-5 (https://doi.org/10.1016%2F0304-4149%2882%2990051-5). ISSN 0304-4149 (https://search.worldcat.org/issn/0304-4149).
24. Luo, Calvin (2022). "Understanding Diffusion Models: A Unified Perspective". arXiv:2208.11970v1 (https://arxiv.org/abs/2208.11970v1) [cs.LG (https://arxiv.org/archive/cs.LG)].
25. Song, Jiaming; Meng, Chenlin; Ermon, Stefano (3 Oct 2023). "Denoising Diffusion Implicit Models". arXiv:2010.02502 (https://arxiv.org/abs/2010.02502) [cs.LG (https://arxiv.org/archive/cs.LG)].

26. Rombach, Robin; Blattmann, Andreas; Lorenz, Dominik; Esser, Patrick; Ommer, Björn (13 April 2022). "High-Resolution Image Synthesis With Latent Diffusion Models". arXiv:2112.10752 (https://arxiv.org/abs/2112.10752) [cs.CV (https://arxiv.org/archive/cs.CV)].
27. Nichol, Alexander Quinn; Dhariwal, Prafulla (2021-07-01). "Improved Denoising Diffusion Probabilistic Models" (https://proceedings.mlr.press/v139/nichol21a.html). *Proceedings of the 38th International Conference on Machine Learning*. PMLR: 8162–8171.
28. Salimans, Tim; Ho, Jonathan (2021-10-06). *Progressive Distillation for Fast Sampling of Diffusion Models* (https://openreview.net/forum?id=TIdIXIpzhoI). The Tenth International Conference on Learning Representations (ICLR 2022).
29. Lin, Shanchuan; Liu, Bingchen; Li, Jiashi; Yang, Xiao (2024). *Common Diffusion Noise Schedules and Sample Steps Are Flawed* (https://openaccess.thecvf.com/content/WACV2024/html/Lin_Common_Diffusion_Noise_Schedules_and_Sample_Steps_Are_Flawed_WACV_2024_paper.html). IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 5404–5411.
30. Dhariwal, Prafulla; Nichol, Alex (2021-06-01). "Diffusion Models Beat GANs on Image Synthesis". arXiv:2105.05233 (https://arxiv.org/abs/2105.05233) [cs.LG (https://arxiv.org/archive/cs.LG)].
31. Ho, Jonathan; Salimans, Tim (2022-07-25). "Classifier-Free Diffusion Guidance". arXiv:2207.12598 (https://arxiv.org/abs/2207.12598) [cs.LG (https://arxiv.org/archive/cs.LG)].
32. Chung, Hyungjin; Kim, Jeongsol; Park, Geon Yeong; Nam, Hyelin; Ye, Jong Chul (2024-06-12). "CFG++: Manifold-constrained Classifier Free Guidance for Diffusion Models". arXiv:2406.08070 (https://arxiv.org/abs/2406.08070) [cs.CV (https://arxiv.org/archive/cs.CV)].
33. Sanchez, Guillaume; Fan, Honglu; Spangher, Alexander; Levi, Elad; Ammanamanchi, Pawan Sasanka; Biderman, Stella (2023-06-30). "Stay on topic with Classifier-Free Guidance". arXiv:2306.17806 (https://arxiv.org/abs/2306.17806) [cs.CL (https://arxiv.org/archive/cs.CL)].
34. Armandpour, Mohammadreza; Sadeghian, Ali; Zheng, Huangjie; Sadeghian, Amir; Zhou, Mingyuan (2023-04-26). "Re-imagine the Negative Prompt Algorithm: Transform 2D Diffusion into 3D, alleviate Janus problem and Beyond". arXiv:2304.04968 (https://arxiv.org/abs/2304.04968) [cs.CV (https://arxiv.org/archive/cs.CV)].
35. Yang, Ling; Zhang, Zhilong; Song, Yang; Hong, Shenda; Xu, Runsheng; Zhao, Yue; Zhang, Wentao; Cui, Bin; Yang, Ming-Hsuan (2022). "Diffusion Models: A Comprehensive Survey of Methods and Applications". arXiv:2206.00364 (https://arxiv.org/abs/2206.00364) [cs.CV (https://arxiv.org/archive/cs.CV)].
36. Shi, Jiaxin; Han, Kehang; Wang, Zhe; Doucet, Arnaud; Titsias, Michalis K. (2024). "Simplified and Generalized Masked Diffusion for Discrete Data". arXiv:2406.04329 (https://arxiv.org/abs/2406.04329) [cs.LG (https://arxiv.org/archive/cs.LG)].
37. Karras, Tero; Aittala, Miika; Aila, Timo; Laine, Samuli (2022). "Elucidating the Design Space of Diffusion-Based Generative Models". arXiv:2206.00364v2 (https://arxiv.org/abs/2206.00364v2) [cs.CV (https://arxiv.org/archive/cs.CV)].
38. Cao, Hanqun; Tan, Cheng; Gao, Zhangyang; Xu, Yilun; Chen, Guangyong; Heng, Pheng-Ann; Li, Stan Z. (July 2024). "A Survey on Generative Diffusion Models" (https://ieeexplore.ieee.org/document/10419041). *IEEE Transactions on Knowledge and Data Engineering*. **36** (7): 2814–2830. doi:10.1109/TKDE.2024.3361474 (https://doi.org/10.1109%2FTKDE.2024.3361474). ISSN 1041-4347 (https://search.worldcat.org/issn/1041-4347).
39. Xu, Yilun; Liu, Ziming; Tian, Yonglong; Tong, Shangyuan; Tegmark, Max; Jaakkola, Tommi (2023-07-03). "PFGM++: Unlocking the Potential of Physics-Inspired Generative Models" (https://proceedings.mlr.press/v202/xu23m.html). *Proceedings of the 40th International Conference on Machine Learning*. PMLR: 38566–38591. arXiv:2302.04265 (https://arxiv.org/abs/2302.04265).
40. Song, Yang; Dhariwal, Prafulla; Chen, Mark; Sutskever, Ilya (2023-07-03). "Consistency Models" (https://proceedings.mlr.press/v202/song23a). *Proceedings of the 40th International Conference on Machine Learning*. PMLR: 32211–32252.

41. Dockhorn, Tim; Vahdat, Arash; Kreis, Karsten (2021-10-06). "Score-Based Generative Modeling with Critically-Damped Langevin Diffusion". arXiv:2112.07068 (https://arxiv.org/abs/2112.07068) [stat.ML (https://arxiv.org/archive/stat.ML)].
42. Liu, Ziming; Luo, Di; Xu, Yilun; Jaakkola, Tommi; Tegmark, Max (2023-04-05). "GenPhys: From Physical Processes to Generative Models". arXiv:2304.02637 (https://arxiv.org/abs/2304.02637) [cs.LG (https://arxiv.org/archive/cs.LG)].
43. Bansal, Arpit; Borgnia, Eitan; Chu, Hong-Min; Li, Jie; Kazemi, Hamid; Huang, Furong; Goldblum, Micah; Geiping, Jonas; Goldstein, Tom (2023-12-15). "Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise" (https://proceedings.neurips.cc/paper_files/paper/2023/hash/80fe51a7d8d0c73ff7439c2a2554ed53-Abstract-Conference.html). *Advances in Neural Information Processing Systems*. **36**: 41259–41282. arXiv:2208.09392 (https://arxiv.org/abs/2208.09392).
44. Gulrajani, Ishaan; Hashimoto, Tatsunori B. (2023-12-15). "Likelihood-Based Diffusion Language Models" (https://proceedings.neurips.cc/paper_files/paper/2023/hash/35b5c175e139bff5f22a5361270fce87-Abstract-Conference.html). *Advances in Neural Information Processing Systems*. **36**: 16693–16715. arXiv:2305.18619 (https://arxiv.org/abs/2305.18619).
45. Lou, Aaron; Meng, Chenlin; Ermon, Stefano (2024-06-06). "Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution". arXiv:2310.16834 (https://arxiv.org/abs/2310.16834) [stat.ML (https://arxiv.org/archive/stat.ML)].
46. Tong, Alexander; Fatras, Kilian; Malkin, Nikolay; Huguet, Guillaume; Zhang, Yanlei; Rector-Brooks, Jarrid; Wolf, Guy; Bengio, Yoshua (2023-11-08). "Improving and generalizing flow-based generative models with minibatch optimal transport" (https://openreview.net/forum?id=CD9Snc73AW). *Transactions on Machine Learning Research*. arXiv:2302.00482 (https://arxiv.org/abs/2302.00482). ISSN 2835-8856 (https://search.worldcat.org/issn/2835-8856).
47. Liu, Xingchao; Gong, Chengyue; Liu, Qiang (2022-09-07). "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow". arXiv:2209.03003 (https://arxiv.org/abs/2209.03003) [cs.LG (https://arxiv.org/archive/cs.LG)].
48. Liu, Qiang (2022-09-29). "Rectified Flow: A Marginal Preserving Approach to Optimal Transport". arXiv:2209.14577 (https://arxiv.org/abs/2209.14577) [stat.ML (https://arxiv.org/archive/stat.ML)].
49. Lipman, Yaron; Chen, Ricky T. Q.; Ben-Hamu, Heli; Nickel, Maximilian; Le, Matt (2023-02-08). "Flow Matching for Generative Modeling". arXiv:2210.02747 (https://arxiv.org/abs/2210.02747) [cs.LG (https://arxiv.org/archive/cs.LG)].
50. Albergo, Michael S.; Vanden-Eijnden, Eric (2023-03-09). "Building Normalizing Flows with Stochastic Interpolants". arXiv:2209.15571 (https://arxiv.org/abs/2209.15571) [cs.LG (https://arxiv.org/archive/cs.LG)].
51. Heitz, Eric; Belcour, Laurent; Chambon, Thomas (2023-05-05). "Iterative α-(de)Blending: a Minimalist Deterministic Diffusion Model". arXiv:2305.03486 (https://arxiv.org/abs/2305.03486) [cs.GR (https://arxiv.org/archive/cs.GR)].
52. "An introduction to Flow Matching · Cambridge MLG Blog" (https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html). *mlg.eng.cam.ac.uk*. Retrieved 2024-08-20.
53. Ho, Jonathan; Saharia, Chitwan; Chan, William; Fleet, David J.; Norouzi, Mohammad; Salimans, Tim (2022-01-01). "Cascaded diffusion models for high fidelity image generation" (https://dl.acm.org/doi/abs/10.5555/3586589.3586636). *The Journal of Machine Learning Research*. **23** (1): 47:2249–47:2281. arXiv:2106.15282 (https://arxiv.org/abs/2106.15282). ISSN 1532-4435 (https://search.worldcat.org/issn/1532-4435).
54. Peebles, William; Xie, Saining (March 2023). "Scalable Diffusion Models with Transformers". arXiv:2212.09748v2 (https://arxiv.org/abs/2212.09748v2) [cs.CV (https://arxiv.org/archive/cs.CV)].
55. Fei, Zhengcong; Fan, Mingyuan; Yu, Changqian; Li, Debang; Huang, Junshi (2024-07-16). "Scaling Diffusion Transformers to 16 Billion Parameters". arXiv:2407.11633 (https://arxiv.org/abs/2407.11633) [cs.CV (https://arxiv.org/archive/cs.CV)].

56. Tevet, Guy; Raab, Sigal; Gordon, Brian; Shafir, Yonatan; Cohen-Or, Daniel; Bermano, Amit H. (2022). "Human Motion Diffusion Model". arXiv:2209.14916 (https://arxiv.org/abs/2209.14916) [cs.CV (https://arxiv.org/archive/cs.CV)].
57. Zhang, Lvmin; Rao, Anyi; Agrawala, Maneesh (2023). "Adding Conditional Control to Text-to-Image Diffusion Models". arXiv:2302.05543 (https://arxiv.org/abs/2302.05543) [cs.CV (https://arxiv.org/archive/cs.CV)].
58. Lugmayr, Andreas; Danelljan, Martin; Romero, Andres; Yu, Fisher; Timofte, Radu; Van Gool, Luc (2022). "RePaint: Inpainting Using Denoising Diffusion Probabilistic Models". arXiv:2201.09865v4 (https://arxiv.org/abs/2201.09865v4) [cs.CV (https://arxiv.org/archive/cs.CV)].
59. Hertz, Amir; Mokady, Ron; Tenenbaum, Jay; Aberman, Kfir; Pritch, Yael; Cohen-Or, Daniel (2022-08-02). "Prompt-to-Prompt Image Editing with Cross Attention Control". arXiv:2208.01626 (https://arxiv.org/abs/2208.01626) [cs.CV (https://arxiv.org/archive/cs.CV)].
60. Zhao, Zheng; Luo, Ziwei; Sjölund, Jens; Schön, Thomas B. (2024). "Conditional sampling within generative diffusion models". arXiv:2409.09650 (https://arxiv.org/abs/2409.09650) [stat.ML (https://arxiv.org/archive/stat.ML)].
61. Wang, Xintao; Xie, Liangbin; Dong, Chao; Shan, Ying (2021). "Real-ESRGAN: Training Real-World Blind Super-Resolution With Pure Synthetic Data" (https://openaccess.thecvf.com/content/ICCV2021W/AIM/papers/Wang_Real-ESRGAN_Training_Real-World_Blind_Super-Resolution_With_Pure_Synthetic_Data_ICCVW_2021_paper.pdf) (PDF). *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, 2021*. International Conference on Computer Vision. pp. 1905–1914. arXiv:2107.10833 (https://arxiv.org/abs/2107.10833).
62. Liang, Jingyun; Cao, Jiezhang; Sun, Guolei; Zhang, Kai; Van Gool, Luc; Timofte, Radu (2021). "SwinIR: Image Restoration Using Swin Transformer" (https://openaccess.thecvf.com/content/ICCV2021W/AIM/papers/Liang_SwinIR_Image_Restoration_Using_Swin_Transformer_ICCVW_2021_paper.pdf) (PDF). *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. International Conference on Computer Vision, 2021. pp. 1833–1844. arXiv:2108.10257v1 (https://arxiv.org/abs/2108.10257v1).
63. Nichol, Alex; Dhariwal, Prafulla; Ramesh, Aditya; Shyam, Pranav; Mishkin, Pamela; McGrew, Bob; Sutskever, Ilya; Chen, Mark (2022-03-08). "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models". arXiv:2112.10741 (https://arxiv.org/abs/2112.10741) [cs.CV (https://arxiv.org/archive/cs.CV)].
64. Ramesh, Aditya; Dhariwal, Prafulla; Nichol, Alex; Chu, Casey; Chen, Mark (2022-04-12). "Hierarchical Text-Conditional Image Generation with CLIP Latents". arXiv:2204.06125 (https://arxiv.org/abs/2204.06125) [cs.CV (https://arxiv.org/archive/cs.CV)].
65. Alammar, Jay. "The Illustrated Stable Diffusion" (https://jalammar.github.io/illustrated-stable-diffusion/). *jalammar.github.io*. Retrieved 2022-10-31.
66. Esser, Patrick; Kulal, Sumith; Blattmann, Andreas; Entezari, Rahim; Müller, Jonas; Saini, Harry; Levi, Yam; Lorenz, Dominik; Sauer, Axel (2024-03-05). "Scaling Rectified Flow Transformers for High-Resolution Image Synthesis". arXiv:2403.03206 (https://arxiv.org/abs/2403.03206) [cs.CV (https://arxiv.org/archive/cs.CV)].
67. Xie, Yiming; Yao, Chun-Han; Voleti, Vikram; Jiang, Huaizu; Jampani, Varun (2024-07-24). "SV4D: Dynamic 3D Content Generation with Multi-Frame and Multi-View Consistency". arXiv:2407.17470 (https://arxiv.org/abs/2407.17470) [cs.CV (https://arxiv.org/archive/cs.CV)].
68. "Imagen: Text-to-Image Diffusion Models" (https://imagen.research.google/). *imagen.research.google*. Retrieved 2024-04-04.
69. Saharia, Chitwan; Chan, William; Saxena, Saurabh; Li, Lala; Whang, Jay; Denton, Emily L.; Ghasemipour, Kamyar; Gontijo Lopes, Raphael; Karagol Ayan, Burcu; Salimans, Tim; Ho, Jonathan; Fleet, David J.; Norouzi, Mohammad (2022-12-06). "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding" (https://proceedings.neurips.cc/paper_files/paper/2022/hash/ec795aeadae0b7d230fa35cbaf04c041-Abstract-Conference.html). *Advances in Neural Information Processing Systems*. **35**: 36479–36494. arXiv:2205.11487 (https://arxiv.org/abs/2205.11487).

70. Chang, Huiwen; Zhang, Han; Barber, Jarred; Maschinot, A. J.; Lezama, Jose; Jiang, Lu; Yang, Ming-Hsuan; Murphy, Kevin; Freeman, William T. (2023-01-02). "Muse: Text-To-Image Generation via Masked Generative Transformers". arXiv:2301.00704 (https://arxiv.org/abs/2301.00704) [cs.CV (https://arxiv.org/archive/cs.CV)].
71. "Imagen 2 - our most advanced text-to-image technology" (https://deepmind.google/technologies/imagen-2/). *Google DeepMind*. Retrieved 2024-04-04.
72. Imagen-Team-Google; Baldridge, Jason; Bauer, Jakob; Bhutani, Mukul; Brichtova, Nicole; Bunner, Andrew; Castrejon, Lluis; Chan, Kelvin; Chen, Yichang (2024-12-13), *Imagen 3*, arXiv:2408.07009 (https://arxiv.org/abs/2408.07009) {{citation}}: |last1= has generic name (help)
73. "Veo" (https://deepmind.google/technologies/veo/). *Google DeepMind*. 2024-05-14. Retrieved 2024-05-17.
74. "Introducing Make-A-Video: An AI system that generates videos from text" (https://ai.meta.com/blog/generative-ai-text-to-video/). *ai.meta.com*. Retrieved 2024-09-20.
75. Singer, Uriel; Polyak, Adam; Hayes, Thomas; Yin, Xi; An, Jie; Zhang, Songyang; Hu, Qiyuan; Yang, Harry; Ashual, Oron (2022-09-29). "Make-A-Video: Text-to-Video Generation without Text-Video Data". arXiv:2209.14792 (https://arxiv.org/abs/2209.14792) [cs.CV (https://arxiv.org/archive/cs.CV)].
76. "Introducing CM3leon, a more efficient, state-of-the-art generative model for text and images" (https://ai.meta.com/blog/generative-ai-text-images-cm3leon/). *ai.meta.com*. Retrieved 2024-09-20.
77. Chameleon Team (2024-05-16). "Chameleon: Mixed-Modal Early-Fusion Foundation Models". arXiv:2405.09818 (https://arxiv.org/abs/2405.09818) [cs.CL (https://arxiv.org/archive/cs.CL)].
78. Zhou, Chunting; Yu, Lili; Babu, Arun; Tirumala, Kushal; Yasunaga, Michihiro; Shamis, Leonid; Kahn, Jacob; Ma, Xuezhe; Zettlemoyer, Luke (2024-08-20). "Transfusion: Predict the Next Token and Diffuse Images with One Multi-Modal Model". arXiv:2408.11039 (https://arxiv.org/abs/2408.11039) [cs.AI (https://arxiv.org/archive/cs.AI)].
79. *Movie Gen: A Cast of Media Foundation Models (https://ai.meta.com/static-resource/movie-gen-research-paper)*, The Movie Gen team @ Meta, October 4, 2024.