**Diffusion models**

**A diagram of a algorithm

Description automatically generated**

Figure 1. from <https://towardsai.net/p/machine-learning/diffusion-models-vs-gans-vs-vaes-comparison-of-deep-generative-models>

Diffusion models is a fairly new class of generative models that was first proposed in 2015 by Sohl-Dickstein et al. [91], which took inspiration from the field of non-equilibrium statistical physics and a phenomenon known as thermodynamic diffusion process [91]. In statistics, “diffusion” refers to the process of transforming a complex distribution to a simple distribution on the same domain. In thermodynamics, “diffusion” refers to the flow of particles from high-density to low-density regions. However, for Sohl-Dickstein et al., “diffusion” means the process of learning a noise-to-data mapping in discrete steps, similar to flow models [91]. They describe their diffusion model as “The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. ” (Sohl-Dickstein et al. (2015), p 1) [91]

Even though diffusion models in machine learning was proposed in 2015, it didn’t catch serious attention until later in 2020 when Ho et al. released “denoising diffusion probabilistic models” [13], which built on and proposed some changes to improve the models proposed by Sohl-Dickstein et al. After OpenAI released further work and improvements on diffusion models in 2021 with “Diffusion Models Beat GANs on Image Synthesis” [10] and “Improved Denoising Diffusion Probabilistic Models” [24], it caught a lot of attention and have been a part of the state-of-the-art of generative models ever since with models like GLIDE, DALLE-2, and imagen form OpenAI, Nvidia and Google.

Simply put, diffusion models iteratively destroy the input data by sequentially adding noise to the images, then learns to undo the destruction of the image by predicting the added noise [4, 13]. The reverse process is learned and completed by a neural network [13]. Diffusion models can be viewed as specific realizations of hierarchical VAEs with some significant differences [4]. The process of destroying the original input, which is known as the forward diffusion process, takes many steps, contains no learnable parameters, and is constructed so that the latent representation of the input is a standard gaussian noise image. The inference of diffusion models is unique since the generation is a product of passing random standard gaussian noise through all the reverse steps to generate a new sample. Even though this is a costly inference compared to other methods like GANs, it produces great results both in terms of quality and diversity [15, 13, 26, 91, 100]. And have some great properties such as easy scalability, stationary training objective and distribution coverage (diversity) [10]

After the releases from Ho et al. (2020) [13] and Dhariwal & Alex Nichol (2021) [10, 24], diffusion models have been further developed [38, 24, 39, 40, 26, 14, 41] to become powerful deep generative models with state-of-the-arts performance in various application such as image synthesis, video generation, and molecule design and have outperformed GANs, VAEs and normalizing flows and partly taken the generative model throne from GANs which have had a long dominance in the field [4, 10, 31, 36, 33, 91, 13, 100, 12, 14, 15, 16].

**Connection to VAE**

VAEs and diffusion model are fairly similar in concepts. Both model classes “destroy” the input into a latent space, then reconstruct the image by learning how it was destroyed [4, 11, 15]. The main differences are that while most VAEs use one step from input to latent representation to reconstruction, diffusion models use many steps in the process. The number of steps used in diffusion models vary, but the number of steps (T) is usually around 1000, meaning the diffusion model uses T steps to destroy the image into a latent representation, then another T steps in the forward process to reconstruct the image. Both models use an encoder-decoder architecture, but the encoder in a diffusion model can be seen as fixed [4].

Another difference is that VAEs compress the images into a latent representation that typically is a lot smaller than the original input. The latent representation of diffusion models is the same size as the original input. This is because diffusion models “destroy” the input by sequentially adding noise to the image that is the same size as the image until it is practically indistinguishable from isotropic gaussian noise, while VAEs “destroy” the input by compressing it [4].

Is diffusion models VAEs? As we will see later, diffusion models technically optimize a variational bound, just as VAEs does. Additionally, Diffusion models and VAEs both have a forward process that transform input data into a latent representation and they both have reverse processes that can sample from a latent space and transform it to a new data. Song et al. (2021) [16] and Kingma et al. (2021) [15] showed that training diffusion models can be viewed as optimizing an infinitely deep hierarchical VAE through the Evidence Lower Bound (ELBO) [4].

But there are key differences that distinguish diffusion models from VAEs.

1. The forward process of diffusion models doesn’t have any learnable parameters.
2. The latent representation of diffusion models (isotropic gaussian noise) contains no information about the original input, which differ from VAEs where the latent representation typically contains some information about the input.
3. The dimension of the latent representation in diffusion models is the same as the original input, while in VAEs the latent representation is often a smaller, compressed representation which means lower dimensionality than the original input
4. Diffusion models uses many steps to destroy the input into a latent representation.

**Connection to normalizing flow**

Normalizing flows is another type of generative models that transform simple probability distributions into complex distributions that is utilized for generation [27, 28, 29]. Normalizing flows differ from VAEs in that they do not perform any dimensionality reduction, and are therefore similar to diffusion model with respect to dimensionality of latent representation being equal to the input dimensionality. Normalizing flow transforms a distribution into a more complex distribution by iteratively applying invertible transformation functions [27, 28, 29]. Applying sequential changes to the input is also similar to the diffusion process. Both diffusion models and normalizing flow transform data to noise. Normalizing flow accomplish this by applying invertible transformations while diffusion models accomplish this by sequentially adding noise to the input.

Both normalizing flow and diffusion models utilize invertible processes. Diffusion model iteratively apply invertible transformation to gradually transform a noise distribution to the data distribution. On the other hand, normalizing flow apply a sequence of invertible transformation to map samples from a base distribution to samples from the target distribution. In both models, the transformations are bijective, which means they are reversible and have a well-defined inversive. This property is essential for tractable density estimation and enables efficient sampling from the model [28, 29].

**Diffusion models**

A diagram of a machine

Description automatically generatedDiffusion models define a Markov chain of diffusion steps to slowly add random noise to data until it is practically isotropic gaussian noise, then another Markov chain is utilized in the diffusion model to learn to reverse the diffusion process to construct desired data samples from the noise [4]. The added noise has the same shape as the input, meaning the latent representation dimensionality is the same as the input which differ from most generative models where the latent representation dimension typically is smaller than the original input. The diffusion process of adding noise and the denoising process utilize multiple steps, which is fundamentally different from other generative models which typically performs both input-to-latent-representation and latent-representation-to-sample in one step each. The intuition of utilizing of multiple such steps is that the model gets the possibility to correct itself when taking the small steps. This makes diffusion models generate great samples at the cost of sampling speed due to the many steps in the inference/sampling process.

The forward process of the diffusion models consists of T steps of sequentially adding gaussian noise to the input. This forward process differ from the typical forward pass of a standard neural network since the diffusion forward process doesn’t utilize a neural network and therefore doesn’t contain any learnable parameters [4]. The objective of the forward process of a diffusion model is to transform the input distribution into a less complex prior distribution, which is most often a standard Gaussian [4, 12]. One remarkably pleasant feature about sequentially adding gaussian noise at each step of the forward process is that the composition or sum of multiple gaussian distributions is itself a gaussian distribution.

The reverse process, or the denoising process, consists of training a neural network to reconstruct/recover the original data through T sequential denoising steps where the neural network learns the noise added in the forward pass and the inverse of this noise so that the model can “denoise” the same steps [4, 12]. Each denoising step utilize the same network, which means each denoising step share parameters. It is this denoising process which correspond to the sampling process in the diffusion model. Generating a new sample with a diffusion model is completed by sampling randomly from the prior distribution that was achieved in the forward process, typically random gaussian noise, then passing the random sampled noise through the reverse process [4, 12].

This process can generate samples with extremely good image quality without any adversarial training which is extremely hard achieve convergence with. However, this comes at the cost of training and sampling speed [12, 4, 10].

**Forward process**

The forward process of a diffusion model is a Markov chain of T sequential steps of iteratively adding noise to the original input. That the forward process is defined as a Markov chain means that each step only depends on the previous step. This process contains no learnable parameters and require no training. The objective of the forward process is to destroy the original input gradually and slowly with small steps by adding noise, usually gaussian noise since it has a couple of nice properties, until the original input has lost its structures [4].

A math equation with arrows pointing to the side

Description automatically generated

Figure 2. How noise is added. given an original image x\_0, the last latent image after T steps x\_1:T is equal to a markov chain from the first to the last image of noise added.

A number of square and square symbols

Description automatically generated

A black background with white text and green and white symbols

Description automatically generated

A black background with arrows pointing to a black background

Description automatically generated

Figure 3. forward process described. markov chain and gaussian noise. B is a fixed variance parameter.

**A math equations and formulas

Description automatically generated with medium confidence**First of all, as the number of steps T goes towards infinity, or become sufficiently large, the final distribution approach an isotropic gaussian distribution. Typically, Both the forward and reverse processes often use thousands of steps (T) for gradual noise injection and during generation for denoising. This makes sampling efficient and easy. Another nice property of utilizing gaussian noise as the destruction steps in the forward process is that the sum of multiple gaussian distributions is itself a gaussian distribution, which indicates that there is no need to apply each forward diffusion action step-by-step. This means that thanks to some reparameterization tricks, a time step t (t E [0, T]) can be sampled with one calculation instead of t step-by-step calculations.

Figure 4. reparametrization trick and new forward diffusion process

Applying the reparameterization trick with gaussian noise modifies the forward diffusion process to randomly sampling t ~ {1, …,T} then applying all the noise steps from 0 to t in one transformation as described with q(x\_t I x\_0). This indicates that it is not necessary to complete a full pass through all small t from 0 to T to sample noisy input.

**A black and white math equation

Description automatically generated with medium confidence**

A black background with white text

Description automatically generated

Figure 5. fully mathematically description of reparameterization trick.

**Backward process**

The forward process contains, as mentioned, no learnable parameters, which indicates that it is not the central part of the generative aspect of the diffusion models. The generative feature of diffusion models is within the so-called backward process or denoising process. Simply put, the backward process reverses the forward process by removing the noise that was added in the forward process. This is done step-by-step over T sequential iterations until the original input is recovered (at least during training, when sampling, the backward process is used to convert random noise into a new sample from the same distribution as the original input). The noise that is being removed is learned and predicted by a model [4]. That means the objective in the backward process is to learn the noise that was added in the forward process and predict how the input would look in the forward process at each timestep t E (0, T). There are three central aspects to create the backward process of the diffusion model:

* 1) A Noise scheduler that sequentially adds noise
* 2) A model that predicts the noise in the image
* 3) A Timestep encoder - a way to encode the current timestep

**Noise scheduler**

The variance B\_t describes how much the input is destroyed at a step. B\_t can be constant or scheduled over the timesteps T. If B\_t is chosen to be changed over the timesteps using a scheduler, then the noise scheduler controls the level of noise that is added to the input during the diffusion process.The noise scheduler in a diffusion model corresponds to a method of adjusting the variance parameter B\_t.

The scheduling of B\_t is important in training a diffusion model since it controls and balance the trade-off between preserving the details of the input and smoothing out the added noise. This is crucial since the input shouldn’t be destroyed to fast or to slow. To slow destruction wouldn’t lead to convergence since the input would not be manipulated into gaussian distribution after the forward diffusion model. To fast destruction of noise would not preserve the features and details of the input long enough, which would impact the difficulty of recover the image in the denoising process. By carefully adjusting the noise schedule, it is possible to control the rate at which the model diffuses the noise and how quickly it converges towards the original image, which would lead to more stable training and better quality of the generated samples. A large B means the pixel distribution of the noise we add is wider but also shifted, this will lead to a more corrupted and noisy image in the next step.

**A graph of a curve

Description automatically generated with medium confidence**Ho et al. (2020) [13] suggested a simple, linear noise scheduler which set the forward variance B\_t equal to a sequence of linearly increasing values between 10^-4 to 0.02. Other noise schedulers have been proposed, for example, Nichol & Dhariwal (2021) [24, 10] proposed a cosine noise scheduler which destroy the input slower than the linear scheduler.The simple linear noise scheduler will be the baseline of this thesis, however, experimenting with implementing a diffusion model utilizing a cosine noise scheduler would be a possibility.

A group of white objects

Description automatically generated

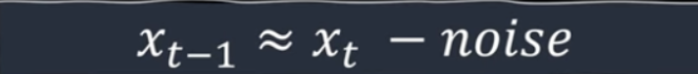
**The model - U-net**

A diagram of a diagram

Description automatically generated with medium confidenceThe model that predicts the noise in the image are typically a U-net [13, 24, 10] and has proven to enhance sample quality [38]. This is because the U-net have multiple benefits in such a problem. Simply put, a U-net is a symmetric neural network architecture, similar to an autoencoder (encoder decoder architecture), where the input and the output have the same dimensionality, which is exactly what is desired in the diffusion model, since the noise have the same size as the input. The U-net is a sort of encoder-decoder architecture that first down sample the input into a bottleneck, then up sample the image back to its original size by using convolutions [10]. Additionally, the U-net utilizes residual connections between the down and up sampling steps that contain the same spatial size, which allows the network to preserve useful and detailed information from earlier layers during up sampling [10]. The residual connections help mitigate the problem of information loss during up sampling

The U-net is designed to capture both local and global features effectively with its down sampling, up sampling and skip connections, which are great attributes in the noise prediction task of the diffusion model. Ho et al. (2020) [13] also proposed group normalization and self-attention blocks to further enhance the performance of the U-net in their denoising diffusion probabilistic model.

In a diffusion task like the one described, where the variance is fixed, the objective of the model is to predict the mean of the gaussian distribution of the noise. This is often referred to as denoising score matching. Fixing the variance in such tasks was proposed by Ho et al. (2022) [13], but it is possible to modify the model to learn the variance of each noise distribution, which openAI and Nichol & Dhariwal (2021) [24) proposed, since they claimed that fixing the variance as done by Ho et al. (2022) [13] was sub-optimal, especially with few diffusion steps T [10].

Getting ahead of ourselves, the next denoised step x\_t-1 given the current denoised step x\_t will be calculated as:

Where the noise is gaussian with the fixed variance B\_t (or B\_t-1) and where the mean of the gaussian noise is predicted by the model

**Timestep encoding**

**A screenshot of a computer

Description automatically generated**Since the neural network (the U-net) shares parameters for each denoising step t, it is necessary for the network to be able to separate between each timestep and execute different denoising steps depending on which timestep t the model is trying to denoise. Separate timesteps have distinct noise added in the forward process, hence the denoising process must separate and denoise distinct patterns for each timestep t. The result of this is the introduction of a conditional variable t in the form of a positional embedding into the model so that it is possible to separate the denoising steps for each timestep [25]. The timestep positional embedding encoding will assign an individual vector to each timestep. In most diffusion models, the timestep positional embedding is utilized using a sinusoidal positional embedding [13, 10]. Applying this timestep positional embedding as input into the model gives the desired application for the diffusion model which is predicting the noise that should be removed in a given timestep t. Now the network will be able to remove distinct noise for given timesteps t while sharing parameters for all denoising steps T.

**Timesteps T**

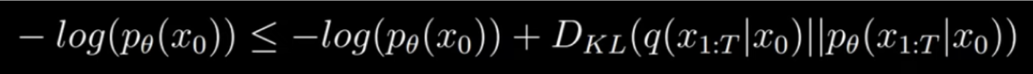
Choosing the number of intermediate diffusion steps T is a crucial part of getting the training of the model to converge and affect sampling time and generation quality [12, 14]. Small T indicates fewer diffusion steps which means the approximation error in the reverse diffusion process increases since the intermediate steps become larger, which indicates greater approximations at each step [12], which is why the baseline for T is usually large, (T = 1000 in Ho et al. (2020) [13])because when T 🡪 infinity, the input will gradually be destroyed until it is a structureless isotropic gaussian, which means the reverse process will be close to a Gaussian [4, 91, 14]. However, the downside with large T is computationally expensive and time consuming both in terms of training and sampling since the reverse diffusion process must iteratively pass through more time step the larger T is [12, 14]. The tradeoff between time/computational efficiency and model/generation quality is important to experiment with when working with diffusion models, since the main disadvantage of diffusion models is the computational inefficiency through the training and sampling speed. Therefore, if it is possible to reduce this inefficiency, it might be worth exploring to limit the computational expense gap between diffusion models and other generative models.

Franzese et al. (2023) [12] provide an evidence-heavy study regarding the diffusion time T and propose that there exist an optimal diffusion time and provide a formal assessment for the best practice for selecting T [12]. The results of the proposed method are improved sampling efficiency and training while containing the sample quality. Simply put, the KL divergence term in the ELBO loss decreases as the diffusion time T increases, which suggests utilizing large T values to maximize the ELBO and improve generation quality [12]. What Franzese et al (2023) [12] propose is to find the sweet spot between the KL terms of the ELBO and the gap G by selecting the best diffusion time T [12].

Song et al. (2022) proposes a method to accelerate the generation speed of diffusion models by utilizing all timesteps T in the forward process, but only sample from some of the diffusion timesteps [14].

**Loss**

Diffusion models are optimized using the variational lower bound, which is similar to the way VAEs are optimized. However, as the theory and understanding of diffusion models expanded, and improvements were proposed, the optimization was altered [10, 13, 15, 24]. This means that the loss for diffusion models went from variational lower bound loss which utilize the Kullback-Leibler divergence (KL divergence), a measure of similarity between two distributions, to something much simpler [15, 13].

Technically, the loss used to optimize diffusion models is the negative log likelihood. However, this is not a computational pleasant term since the probability distribution of the final output x\_0 in the reverse process is not easily computed because it depends on all the previous timesteps, which would imply storing all variables from all the previous diffusion steps. To simplify this, the loss function in diffusion models is the variational lower bound of the negative log likelihood [13, 10].

This expression is not computational pleasant either, since it still utilizes the probability distribution of x\_0. Some mathematical computations and simplifications of the KL divergence term present the opportunity to cancel out the terms containing x\_0.

A black background with white text

Description automatically generated

A black background with white text

Description automatically generatedA black background with white text

Description automatically generated

**🡪**

The variational lower bound of the negative log likelihood is set. But applying bayes rules and a lot of log rules simplifies the variational lower bound further.

[**https://www.youtube.com/watch?v=a4Yfz2FxXiY&ab\_channel=DeepFindr**](https://www.youtube.com/watch?v=a4Yfz2FxXiY&ab_channel=DeepFindr)

or <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

A group of mathematical equations

Description automatically generated

A black background with white text

Description automatically generatedA black background with white text

Description automatically generated

A black background with white text

Description automatically generated

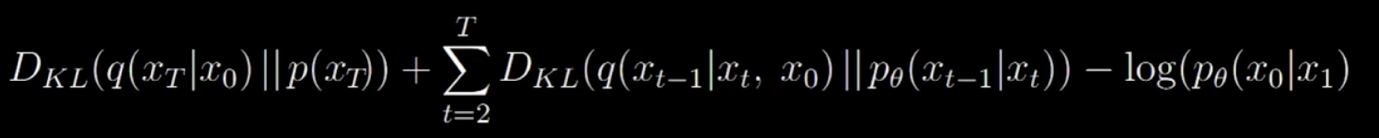
A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

A black background with math equations

Description automatically generatedA black background with white text

Description automatically generatedA black background with white text

Description automatically generated

A black background with white text

Description automatically generated

A black rectangular with white text and yellow text

Description automatically generated

**🡪**

All this work shows that what the variational lower bound essentially are subtracting random scaled noise from x\_t. Ho et al. (2020) [13] ignored the weighted term and decided to use a simple means squared error between the actual mean of the added noise in the forward process and the predicted mean in the backward process [13]. Kingma et al. (2021) [15] also show that the variational lover bound can be simplified to a surprisingly simple term. That means that the final loss function is defined as calculating the L2 difference between the predicted loss and the actual noise in the image.

**Training algorithm**

With all the necessary pieces, it is time to understand how the diffusion model is trained.

**A math equations and numbers

Description automatically generated with medium confidence**

The training process consists of taking an input x\_0, sampling a random timestep t from (1, T).

The key part of the training process is that the reverse process approximates the forward process as closely as possible [4]. This is accomplished by minimizing the KL divergence between the forward and backward process [4].

**Sampling**

Once the model is trained, generating new samples is fairly straightforward and simple; a random noise (from the prior distribution, which typically is gaussian) vector x\_T of the same size as the original input is generated. Then for each timestep T, noise is removed from gradually by passing x\_T through the model T times along with the positional embedding encoding of timestep t to remove the correct noise at the current timestep. This is iteratively completed until t = 0 and x\_T is completely denoised [16]. This method is also known as Langevin dynamics [16].

**Downsides of diffusion models**

Diffusion models have proved to perform extremely well on generative tasks in the last years and outperforms other models in the same field [4, 10, 31, 36, 33, 91, 13, 100, 12, 14, 15, 16]. However, the performance and results come with some consequences and drawbacks [4. 10, 14].

The big weakness of diffusion models is that it is computational expensive both in terms of training the model and sampling [4, 10, 14]. Song et al (2020) [14] illustrates the sampling inefficiency with a concrete example “, it takes around 20 hours to sample 50k images of size 32 × 32 from a DDPM, but less than a minute to do so from a GAN on an Nvidia 2080 Ti GPU” (Song et al., 2020, p1) [14]. This implies a x1200 sampling time with diffusion models compared to GANs. The sampling inefficiency is a result of the step-by-step iterative Markov chain in the denoising process where each denoising timestep t requires a pass through the neural network [4].

The inefficiency of diffusion models is a working field that are trying to be improved so that the quality, diversity, and performance of diffusion model can be further utilized [12, 14]. Compared to the extremely fast and well-performing GANs, the time consumption and computational expensiveness of diffusion models leave the choice of generative models open in terms of the beneficial features and downsides of the two models in a tradeoff that must be carefully evaluated for the choice of model [10].

One solution to the inefficiency of diffusion model combines a diffusion model with the encoder-decoder architecture that is typical in autoencoders, VAEs and GANs and is known as latent diffusion.

**Latent diffusion**

The idea of latent diffusion is to pretrain an encoder-decoder architecture with a prior latent distribution (gaussian) between the two architecture components, similar to a VAE or a GAN, and then training the latent space further using a diffusion model to bring the latent encoding even closer to the prior distribution. The objective with this approach is to optimize the latent space representation [42].

The resulting diffusion process is compressed into a much simpler version that is applying the diffusion processes into a smaller latent space (of size (1,300) for example) instead of an image/pixel space (64, 64). The outcome is much faster training, diffusion process and sampling while managing to optimize the latent space [42].

The downside of latent diffusion is that it requires a pretrained encoder-decoder architecture, and that the latent diffusion process optimizes the latent space of that specific encoder-decoder model. This means that if the VAE or GAN that is trained produces sparse, images, the diffusion model might be able to improve the quality of the generated samples, but the main features of the generated samples are from the pretrained encoder-decoder architecture.

**Diagram of a diagram of a flow of light

Description automatically generated**

**A diagram of a diagram of a wave

Description automatically generated**