High School Student Institute of Mathematics and Informatics

Summer Research School
Symposium
2023

# The state of Latent Diffusion models

**Author(s):**

**Erik Paskalev NPMG "Academician Lyubomir Chakalov"**
erik.paskalev@gmail.com

**Scientific Advisor(s):**

**Delyan Boychev**
**High School of Mathematics and Natural Sciences**
**Veliko Tarnovo, Bulgaria**
delyan.boychev05@gmail.com

**Kostadin Delchev**
**Institute of Mathematics and Informatics**
**Bulgarian Academy of Sciences**
math_k_delchev@yahoo.com

## Abstract

In this paper we provide a summarization of the architecture of Latent Diffusion Models (LDMs) like Stable diffusion, DALL-E 2, Midjourney, and also one of the most popular and effective fine-tuning techniques Low-Rank Adaptation (LoRA) with some of its variants.

# 1 Introduction

We will explore the theory behind Stable Diffusion, by building up from the most basic Diffusion model to the state-of-the-art and see the recent improvements and future potential, alongside the challenges that lie ahead.
The concepts that will be required prerequisites for full comprehension of this paper are: understanding a solid foundational understanding of Deep Learning, Computer Vision, Natural language processing (e.g Multi-layer Perceptron, Normalization, Regularization, Convolutional neural networks, Generative Adversarial Networks, Auto-encoders, Attention, Transformers), basics of Probability theory and statistics (e.g Probability distributions, Conditional probability, Expected value, Bayesian statistics). Other useful knowledge but not fully necessary includes Linear algebra (e.g. vectors, dot product, span, matrices), basics of calculus (e.g. derivatives, integrals), and Markov chains.

# 2 Diffusion models

Diffusion models have taken over modern-day computer vision, achieving state of the art in a variety of tasks. They reach similar image quality to GANs while not suffering from mode collapse. Diffusion models have 2 processes, one forward noising process and one reverse denoising process see Figure 1.
The model takes an input image $x_0$ and runs it through the Forward Diffusion process which is modeled as a conditional probability $q(x_t|x_{t-1})$. It consists of a step-by-step process where at each step we add a fixed amount of Gaussian noise to the image see Figure 2, this is repeated until the image becomes complete noise $x_T$. Then we take the resulting noise and run it step by step through the Reverse process, modeled as $p_\theta(x_{t-1}|x_t)$, where at each step the goal is to predict the noise added to the image $x_{t-1}$. This is repeated until we reach $x_0$.
Unlike the noising process the denoising isn't fixed, $p_\theta$ is a learned function, e.g. a neural network, with parameters $\theta$. In practice [8] proved it is better to, instead of directly learning $p_\theta(x_{t-1}|x_t)$, to learn the noise added to $x_{t-1}$ modeled as $\epsilon_\theta(x_t, t)$ where t is the current time step, and derive $x_{t-1}$ from this estimate. The training process of this neural network involves taking a random image $x_0$, time step $t$, noise $\epsilon$ and optimizing the Mean-squared error loss between the predicted noise and the true noise $||\epsilon_\theta(x_t, t) - \epsilon||^2$. Based on this definition, the current model isn't useful it takes as input an image and returns a at best slightly imperfect and blurry image and at worse nonsense. The key lies in the fact that during inference specifically for image synthesis we skip the Forward Diffusion process entirely and directly pass noise to the Reverse Process. Meaning unlike before the model no longer needs the original image as input thus we can generate completely novel images.
With this current architecture, there are two big problems the first being the ability to control what kind of images the model generates. The only form of control we currently have is the noise we give as input at inference and the training data. Controlling with the noise is a fruitless endeavor as we have no way of knowing what noise leads to what images and controlling with the training data is somewhat effective but very computationally expensive and requires the model to be retrained from scratch on a very specific hand-selected dataset for every desired change in the kinds of images it generates.
The second problem is the Inference time if we look at the effect of the number of total time steps $T$. The greater the number of steps the smaller the added noise will be between each step and the easier it will be for the reverse process to estimate each step resulting in greater image quality the big trade-off however is Inference time since during inference you have to run the reverse process for each step sequentially so if $T = 1000$ the reverse process and thus the model will run 1000 times making this the biggest weakness of current Diffusion architecture as a whole.

## 2.1 Forward Process

This section and the next will cover some of the details that were glossed over in the main explanation. [8]. First is the choice of the fixed noise that is added at each time step, which is determined by a so-called scheduler. More specifically it determines how much Gaussian noise is added at each step with there being two main choices: a Linear schedule and a

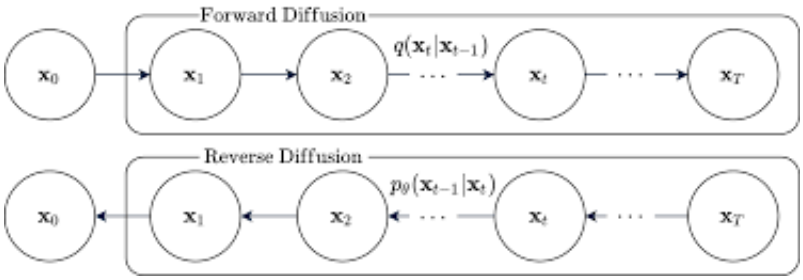


Figure 1: the Diffusion process, first the image is iteratively passed through the Forward Diffusion then passed back through the Reverse Diffusion at each step predicting the image $x_{t-1}$ until it reaches $x_0$. source: [3]


Figure 2: the Forward Diffusion process, we add noise to the image at each time step. The resulting image gets noisier until it is unrecognizable. Image source:[2]

cosine schedule[13]. The difference can be observed in see Figure 3. Lastly, the forward process doesn't need to be computed iteratively, it can be directly computed in one step $q(x_t, x_0)$ as proved by [8]. Lastly, a more general way to describe this part of the model is as a series of steps through a Markov chain that has only one transition at each state.

### 2.2 Reverse Process

The reverse process is the most complicated part of the entire model. First, we need to decide the exact neural net architecture that we are going to use. The original paper proposes the U-Net[17] see Figure 4, more specifically an improved version of the Attention U-Net[14]. The U-Net is a fully convolutional network, it consists of two main parts, a contracting path, and an expanding path. The contracting path consists of multiple applications of a downsampling block which is made up of two 3x3 unpadded convolutions and ReLU activations and one 2x2 max pooling layer with stride 2. With each down-sampling step, the number of feature channels doubles. The expansive path is the same however the 2x2 max pooling is replaced by 2x2 up-sampling[17]. Each pair of blocks equivalent size blocks is connected by a residual connection that sends the feature learned from the down-sampling block to the up-sampling block and alongside the output of the lower block is concatenated and passed through an attention gate. Attention gates filter the noisy parts of the input

Figure 3: The Linear scheduler(top) adds noise faster than the cosine scheduler(bottom) thus destroying information faster making it worse than the cosine scheduler source:[2]

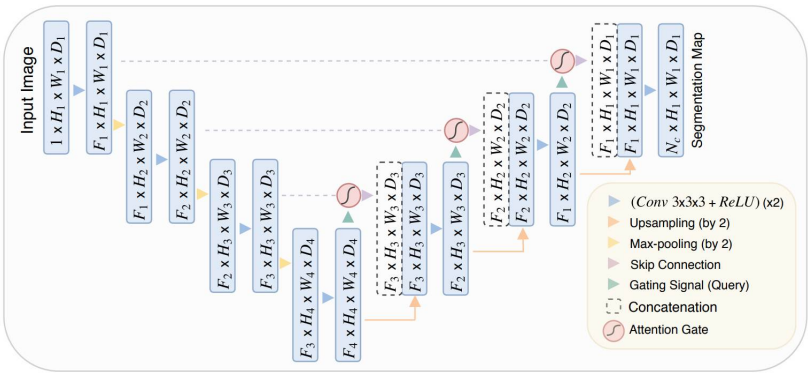


Figure 4: A visual representation of the architecture source:[4]

and focus the attention of the model on the local lower-level features of each region of the image[14].

As proven by [8] the definition of the optimization task of the model has three equivalent definitions:

1. Predict the original image $x_0$
2. Predict the mean of the next image $x_{t-1}$
3. Predict the mean of the noise added to $x_{t-1}$

Option one is rejected in [19] and options two and three are equivalent just parameterized differently as shown well in [12]. In the pilot paper, the authors went with the latter of the two options. The last major decision is the loss function the original paper start with the standard for VAEs the Variational lower bound but derived the much simpler and better in practice mean-squared error loss between the predicted noise and the true noise $||\epsilon_\theta(x_t, t) - \epsilon||^2$ later it was improved by predicting not only the mean of the noise but also it's variance[13]. It combines the MSE with the VLB multiplied by some very small constant. This is because the variance primarily influences image quality at lower values of $T$[13], meaning the improvement of this loss function comes when $T$ is small $< 50$.

## 3 Conditioning and Guidance

Conditioning substantially improves the controllability problem of diffusion models see Figure 5. It starts by taking an input text prompt that is tokenized and turned into positional embeddings for each token. It is then passed through a Transformer block and converted into an image then it is concatenated with the time step embedding $t$ and it is inserted in the model at the start with the noise, at the normalization layers, and at the Attention gates[16]. The conditioning doesn't need to be text recently much research into image-based and multimodal conditioning[5].

Guidance is a hyperparameter that tells the model how strictly it must take the conditioning input into account. Without this the model can sometimes uncontrollably ignore the input. Combining these two ideas is the backbone of modern image synthesis models.

## 4 Latent Diffusion models

The main idea of Latent diffusion models is to introduce an autoencoder that downscales the input image into a lower dimensionality see Figure 6 and rather than working with the original images the model now operates in lower dimensional latent space. This significantly improves the performance of the model while retaining competitive image quality[16]. There are 3 things that are added to the model. First is the encoder at the very start of the model, it takes the original image and converts it into the downsampled latent space see Figure 6,

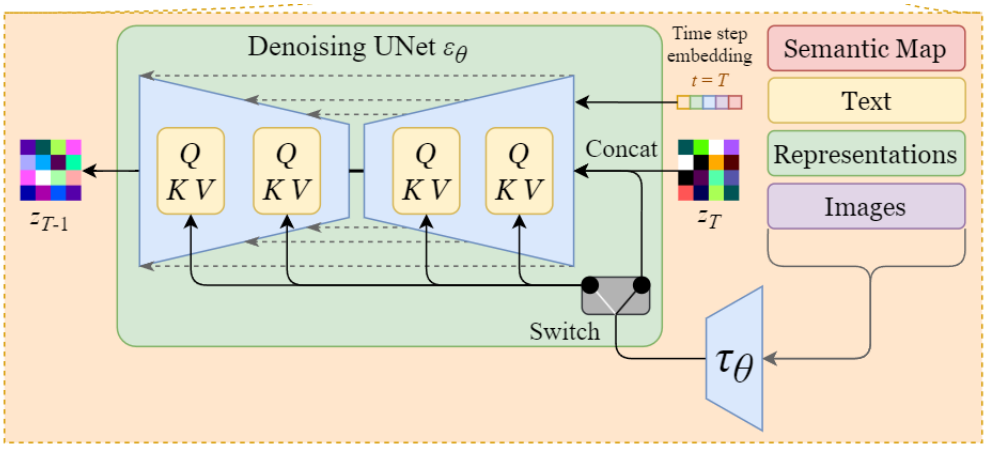


Figure 5: Diagram of the current reverse denoising process, $t_\theta$ should be ignored in this diagram it doesn't serve a purpose in this step. source:[1]

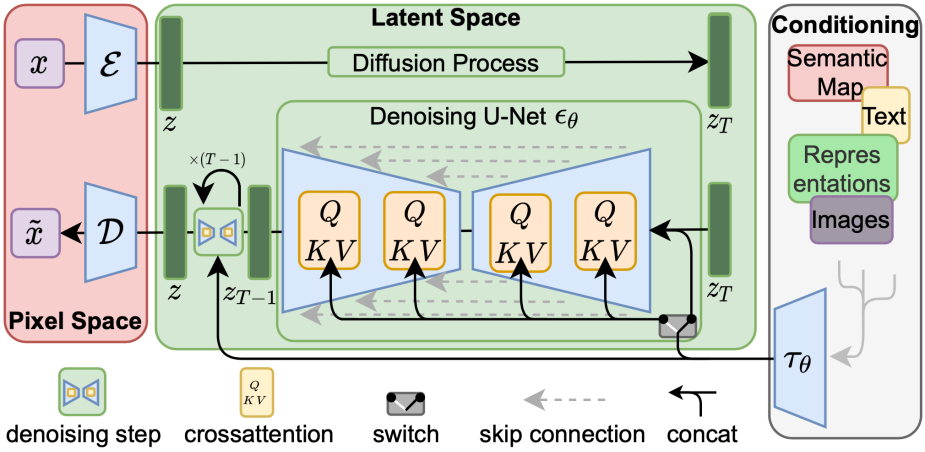


Figure 6: $t_\theta$ should be ignored in this diagram it doesn't serve a purpose until the last step source:[16]

second is the decoder that sits at the end and converts it back to an image. Last is the addition of $\tau_\theta$ - a cross-attention transformer that converts the conditioning input to the model latent space. This is at the present moment the state-of-the-art architecture for image synthesis. Despite the big speedups, they still lag behind GANs in inference speed[16].

## 5 Low-Rank Adaptation

Low-Rank Adaptation aka. LoRA is a parameter-efficient fine-tuning technique, originally created for Large Language models[10]. Its use in diffusion models is to insert a specific concept into the diffusion model. For comparison regular fine-tuning is the same process as regular training, except the initial model weights are taken from the already trained model and the training data is highly specialized to be about one concept. The main problem with this approach is that it is just as intensive computationally as regular training it requires that the entire model be stored in the memory and gradients to be calculated for all parameters. Because of this working with fine-tuned models can be difficult in practice as they are expensive to deploy and share with others. LoRA works by changing what parameters it works on. Rather than directly updating the original model parameters, which for a given layer are a matrix $W_0$, it freezes them and updates an auxiliary matrix $\Delta W$. During

fine-tuning and inference, the original model weights and auxiliary weights are added to produce the final model weights. This is combined with the idea that the "intrinsic rank" of $\Delta W \in \mathbb{R}^{d \times k}$ is low meaning we decompose it into two matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ such that the rank $r$ is smaller than $d$ and $k$ $\Delta W = BA$. This way we have a hyperparameter $r$ that we can tune depending on the task. In the most extreme case $r = 1$, depending on the size of the matrix the number of parameters is decreased by $\times 10000$. When using LoRA in practice [10] found that it is optimal to apply it only to the Attention layers, since those layers are most sensitive to change. The reason for this is that while LoRA does save significant amounts of memory it is slower than normal fine-tuning because the number of gradients that need to be computed is greater, by looking at a small subsection of the model most gradients aren't computed. When we look at the latent diffusion model there are two places where LoRA is implemented: the text encoder and the attention gates in the U-Net. Comparing LoRA to other parameter-efficient fine-tuning techniques, the main alternatives are prefix tuning[11] and adapter layers[9]. The former involves allocating a certain prefix of the input tokens to "steer" the model in the right direction. This method has been observed as difficult to optimize and it reduces the maximum size an input can be[10]. The latter attaches a small number of layers (typically MLPs) at the end of the model and optimizes only those. This however suffers from an increase in latency[10], as modern-day architectures rely on parallelization to achieve high performance these layers are a bottleneck to inference. All of these techniques however including LoRA need very well handpicked datasets.

## 6 Future work

There is currently a lot of active research into Latent Diffusion models, one of which is C-LoRA[18]. It aims to change the way diffusion models learn by doing what they call "Continual Diffusion". It involves learning each concept individually, using 50-100 very well-selected images. The main difficulty of this approach is preventing catastrophic forgetting. They attempt to solve this by localizing each concept to a specific subset of the model. The next step in Diffusion models is the generation of full videos, they show promising results achieving state-of-the-art [6] by first training on images and then "video fine-tuning" the model by introducing "temporal" neural network layer interleaved with "spatial" layers such that each generated image is "temporally consistent".

## 7 Conclusion

Latent Diffusion models have come to the forefront of computer vision tasks with their image quality and diversity, combined with Low-Rank Adaptation high-quality image synthesis has never been as widespread and they provide a promising path for future research.

## References

[1] Attention UNET and its Implementation in TensorFlow. https://medium.com/@steinsfu/stable-diffusion-clearly-explained-ed008044e07e.

[2] A gentle introduction to Dance diffusion. https://wandb.ai/wandb_gen/audio/reports/A-Gentle-Introduction-to-Dance-Diffusion--VmlldzoyNjg1Mzky.

[3] An introduction to Diffusion Probabilistic Models. https://ayandas.me/blog-tut/2021/12/04/diffusion-prob-models.html.

[4] Attention UNET and its Implementation in TensorFlow. https://idiotdeveloper.com/attention-unet-and-its-implementation-in-tensorflow/.

[5] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 843–852, June 2023.

[6] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22563–22575, June 2023.

[7] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[9] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

[10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[11] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[12] Calvin Luo. Understanding diffusion models: A unified perspective, 2022.

[13] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[14] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018.

[15] OpenAI. Gpt-4 technical report, 2023.

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora, 2023.

[19] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.