DIT Model Report

[Model Overview 2](#_Toc164725352)

[**1.** The autoencoder (VAE) 3](#_Toc164725353)

[**2**. The U-Net 3](#_Toc164725354)

[**3.** The Text-encoder 3](#_Toc164725355)

[**Why is latent diffusion fast and efficient?** 4](#_Toc164725356)

[**Stable Diffusion during inference** 4](#_Toc164725357)

# Model Overview

DIT ( Diffusion transformer model), Stable Diffusion is based on a particular type of diffusion model called **Latent Diffusion**, proposed in High-Resolution Image Synthesis with Latent Diffusion Models.

General diffusion models are machine learning systems that are trained to denoise random gaussian noise step by step, to get to a sample of interest, such as an image.

Diffusion models have shown to achieve state-of-the-art results for generating image data. But one downside of diffusion models is that the reverse denoising process is slow. In addition, these models consume a lot of memory because they operate in pixel space, which becomes unreasonably expensive when generating high-resolution images. Therefore, it is challenging to train these models and also use them for inference.

Latent diffusion can reduce the memory and compute complexity by applying the diffusion process over a lower dimensional latent space, instead of using the actual pixel space. This is the key difference between standard diffusion and latent diffusion models: **in latent diffusion the model is trained to generate latent (compressed) representations of the images.**

There are three main components in latent diffusion.

1. An autoencoder (VAE).
2. A U-Net.
3. A text-encoder, e.g. CLIP's Text Encoder.

# **1.** The autoencoder (VAE)

The VAE model has two parts, an encoder and a decoder. The encoder is used to convert the image into a low dimensional latent representation, which will serve as the input to the U-Net model. The decoder, conversely, transforms the latent representation back into an image.

During latent diffusion training, the encoder is used to get the latent representations (latents) of the images for the forward diffusion process, which applies more and more noise at each step. During inference, the denoised latents generated by the reverse diffusion process are converted back into images using the VAE decoder. As we will see during inference we **only need the VAE decoder**.

# **2**. The U-Net

The U-Net has an encoder part and a decoder part both comprised of ResNet blocks. The encoder compresses an image representation into a lower resolution image representation and the decoder decodes the lower resolution image representation back to the original higher resolution image representation that is supposedly less noisy. More specifically, the U-Net output predicts the noise residual which can be used to compute the predicted denoised image representation.

To prevent the U-Net from losing important information while downsampling, short-cut connections are usually added between the downsampling ResNets of the encoder to the upsampling ResNets of the decoder. Additionally, the stable diffusion U-Net is able to condition its output on text-embeddings via cross-attention layers. The cross-attention layers are added to both the encoder and decoder part of the U-Net usually between ResNet blocks.

# **3.** The Text-encoder

The text-encoder is responsible for transforming the input prompt, e.g. "A crack in a wall" into an embedding space that can be understood by the U-Net. It is usually a simple transformer-based encoder that maps a sequence of input tokens to a sequence of latent text-embeddings.

Inspired by [Imagen](https://colab.research.google.com/corgiredirector?site=https%3A%2F%2Fimagen.research.google%2F), Stable Diffusion does **not** train the text-encoder during training and simply uses an CLIP's already trained text encoder, [CLIPTextModel](https://colab.research.google.com/corgiredirector?site=https%3A%2F%2Fhuggingface.co%2Fdocs%2Ftransformers%2Fmodel_doc%2Fclip%23transformers.CLIPTextModel" \t "_blank).

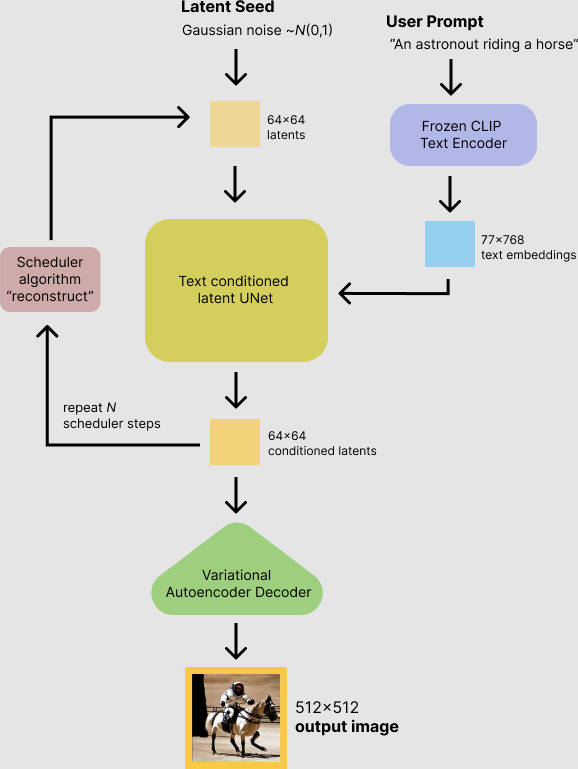
# **Why is latent diffusion fast and efficient?**

Since the U-Net of latent diffusion models operates on a low dimensional space, it greatly reduces the memory and compute requirements compared to pixel-space diffusion models. For example, the autoencoder used in Stable Diffusion has a reduction factor of 8. This means that an image of shape (3, 512, 512) becomes (3, 64, 64) in latent space, which requires 8 × 8 = 64 times less memory.

This is why it's possible to generate 512 × 512 images so quickly, even on 16GB Colab GPUs!

# **Stable Diffusion during inference**

Putting it all together, let's now take a closer look at how the model works in inference by illustrating the logical flow.



The stable diffusion model takes both a latent seed and a text prompt as an input. The latent seed is then used to generate random latent image representations of size 64×64 where as the text prompt is transformed to text embeddings of size 77×768 via CLIP's text encoder. Next the U-Net iteratively denoises the random latent image representations while being conditioned on the text embeddings. The output of the U-Net, being the noise residual, is used to compute a denoised latent image representation via a scheduler algorithm. Many different scheduler algorithms can be used for this computation, each having its pros and cons. For Stable Diffusion, we recommend using one of:

• PNDM scheduler (used by default).

• K-LMS scheduler.

• Heun Discrete scheduler.

• DPM Solver Multistep scheduler.

This scheduler is able to achieve great quality in less steps. You can try with 25 instead of the default 50!

Theory on how the scheduler algorithm function is out of scope for this notebook, but in short one should remember that they compute the predicted denoised image representation from the previous noise representation and the predicted noise residual. For more information, we recommend looking into Elucidating the Design Space of Diffusion-Based Generative Models

The denoising process is repeated ca. 50 times to step-by-step retrieve better latent image representations. Once complete, the latent image representation is decoded by the decoder part of the variational auto encoder.