

STABLE DIFFUSION ART < [HTTPS://STABLE-DIFFUSION-ART.COM/](https://stable-diffusion-art.com/)>



How does Stable Diffusion work?

Updated July 23, 2023 · By [Andrew](#) · Categorized as [Tutorial](#) ·

Tagged [image-to-image](#), [models](#), [text-to-image](#) ·

[21 Comments < https://stable-diffusion-art.com/how-stable-diffusion-work/#comments>](#)

Stable Diffusion is a deep-learning model. We will dig deep into understanding how it works under the hood.

Why do you need to know? Apart from being a fascinating subject in its own right, some understanding of the inner mechanics will make you a better artist. You can use the tool correctly to achieve results with higher precision.

How does text-to-image differ from image-to-image? What's CFG value? What's denoising strength? You will find the answer in this article.

Let's dive in.

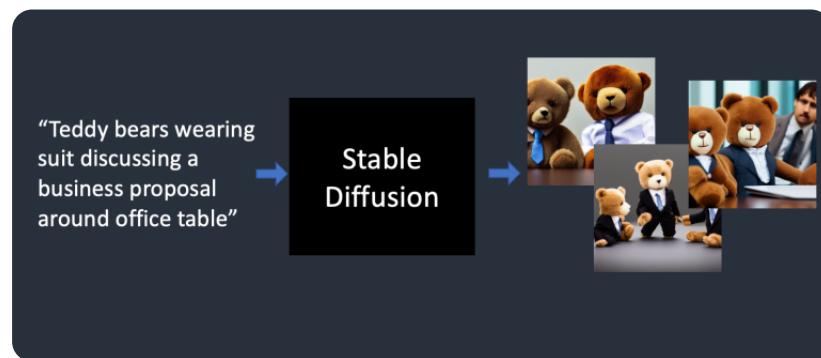
Contents [hide]

- What can Stable Diffusion do?
- Diffusion model
- Forward diffusion
- Reverse diffusion

How training is done
 Reverse diffusion
 Stable Diffusion model
 Latent diffusion model
 Variational Autoencoder
 Image resolution
 Why is latent space possible?
 Reverse diffusion in latent space
 What is a VAE file?
 Conditioning
 Text conditioning (text-to-image)
 Tokenizer
 Embedding
 Feeding embeddings to noise predictor
 Cross-attention
 Other conditionings
 Stable Diffusion step-by-step
 Text-to-image
 Noise schedule
 Image-to-image
 Inpainting
 Depth-to-image
 What is CFG value?
 Classifier guidance
 Classifier-free guidance
 CFG value
 Stable Diffusion v1 vs v2
 Model difference
 Training data difference
 Outcome difference
 Some interesting reads

What can Stable Diffusion do?

In the simplest form, Stable Diffusion is a **text-to-image model**. Give it a **text prompt**. It will return an image matching the text.



Stable diffusion turns text prompts into images.

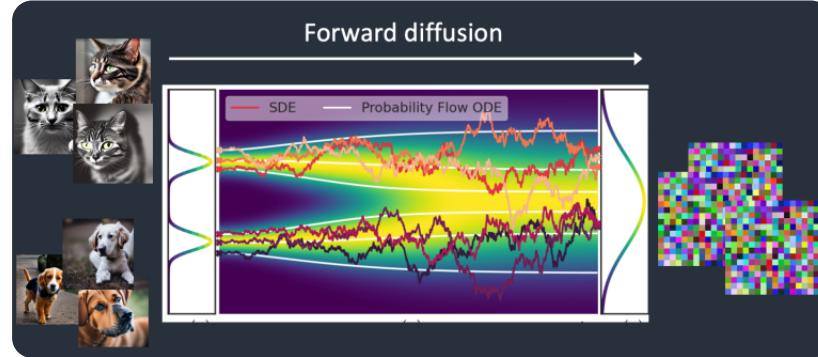
Diffusion model

Stable Diffusion belongs to a class of deep learning models called **diffusion models**. They are generative models, meaning they are designed to generate new data similar to what they have seen in training. In the case of Stable Diffusion, the data are images.

Why is it called the diffusion model? Because its math looks very much like diffusion in physics. Let's go through the idea.

Let's say I trained a diffusion model with only two kinds of images: cats and dogs. In the figure below, the two peaks on the left represent the groups of cat and dog images.

Forward diffusion

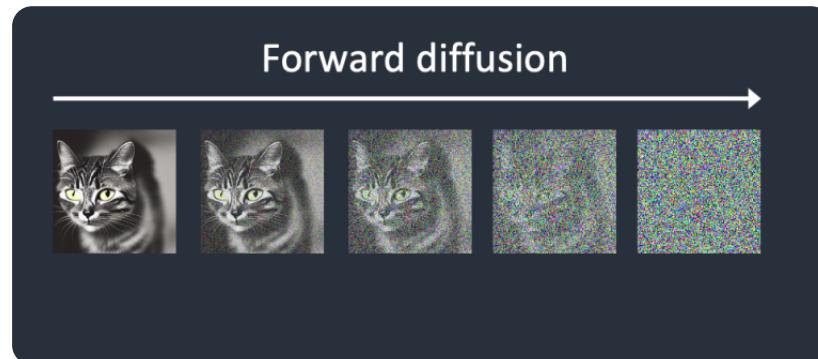


Forward diffusion turns a photo into noise. (Figure modified from [this article <https://arxiv.org/abs/2011.13456>](#))

A **forward diffusion** process adds noise to a training image, gradually turning it into an uncharacteristic noise image. The forward process will turn any cat or dog image into a noise image. Eventually, you won't be able to tell whether they are initially a dog or a cat. (This is important)

It's like a drop of ink fell into a glass of water. The ink drop **diffuses** in water. After a few minutes, It randomly distributes itself throughout the water. You can no longer tell whether it initially fell at the center or near the rim.

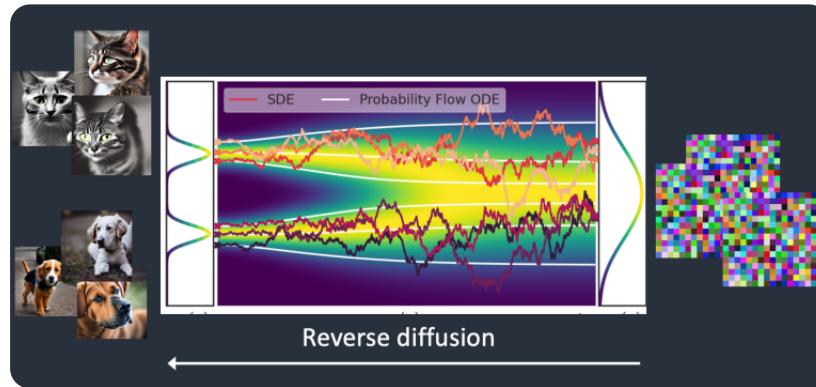
Below is an example of an image undergoing forward diffusion. The cat image turns to random noise.



Forward diffusion of a cat image.

Reverse diffusion

Now comes the exciting part. What if we can reverse the diffusion? Like playing a video backward. Going backward in time. We will see where the ink drop was initially added.



The reverse diffusion process recovers an image.

Starting from a noisy, meaningless image, **reverse diffusion** recovers a cat **OR** a dog image. This is the main idea.

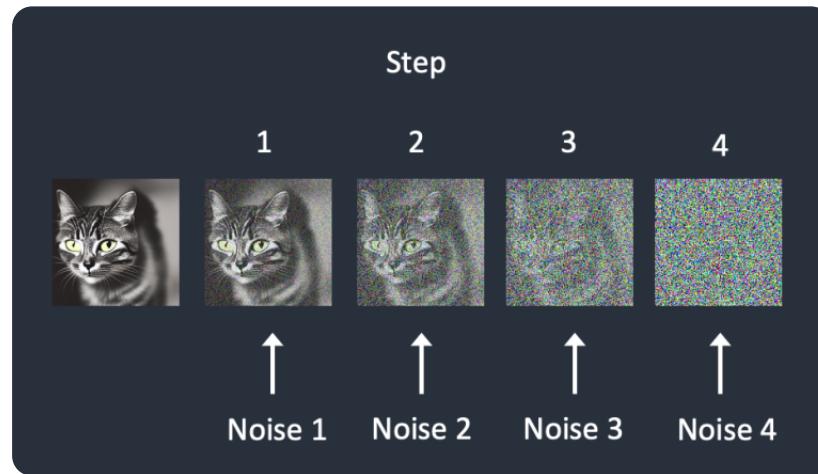
Technically, every diffusion process has two parts: (1) **drift** and (2) **random** motion. The reverse diffusion drifts towards either cat OR dog images but nothing in between. That's why the result can either be a cat or a dog.

How training is done

The idea of reverse diffusion is undoubtedly clever and elegant. But the million-dollar question is, "How can it be done?"

To reverse the diffusion, we need to know how much noise is added to an image. The answer is teaching a **neural network model to predict the noise added**. It is called the **noise predictor** in Stable Diffusion. It is a **U-Net model** <<https://en.wikipedia.org/wiki/U-Net>> . The training goes as follows.

1. Pick a training image, like a photo of a cat.
2. Generate a random noise image.
3. Corrupt the training image by adding this noisy image up to a certain number of steps.
4. Teach the noise predictor to tell us how much noise was added. This is done by tuning its weights and showing it the correct answer.



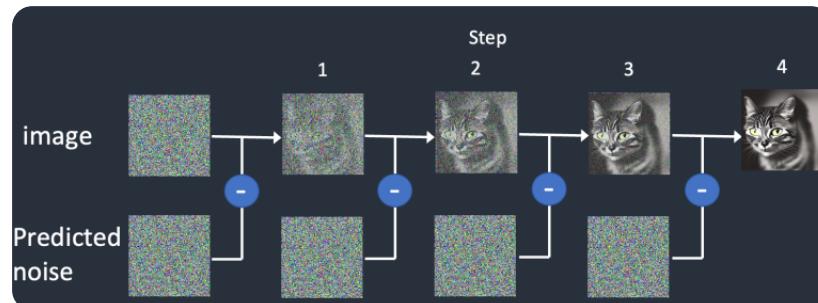
Noise is sequentially added at each step. The noise predictor estimates the total noise added up to each step.

After training, **we have a noise predictor capable of estimating the noise added to an image.**

Reverse diffusion

Now we have the noise predictor. How to use it?

We first **generate a completely random image and ask the noise predictor to tell us the noise**. We then subtract this estimated noise from the original image. Repeat this process a few times. You will get an image of either a cat or a dog.



Reverse diffusion works by subtracting the predicted noise from the image successively.

You may notice we have no control over generating a cat or dog's image. We will address this when we talk about conditioning. For now, image generation is **unconditioned**.

You can read more about reverse diffusion sampling and samplers in [this article < https://stable-diffusion-art.com/samplers/>](https://stable-diffusion-art.com/samplers/).

Stable Diffusion model

Now I need to tell you some bad news: What we just talked about is NOT how Stable Diffusion works! The reason is that the above diffusion process is in image space. It is computationally very, **very slow**. You won't be able to run on any single GPU, let alone the crappy GPU on your laptop.

The image space is enormous. Think about it: a 512×512 image with three color channels (red, green, and blue) is a 786,432-dimensional space! (You need to specify that many values for ONE image.)

Diffusion models like Google's **Imagen** <<https://Imagen.research.google/>> and Open AI's **DALL-E** <<https://openai.com/dall-e-2/>> are in pixel space. They have used some tricks to make the model faster but still not enough.

Latent diffusion model

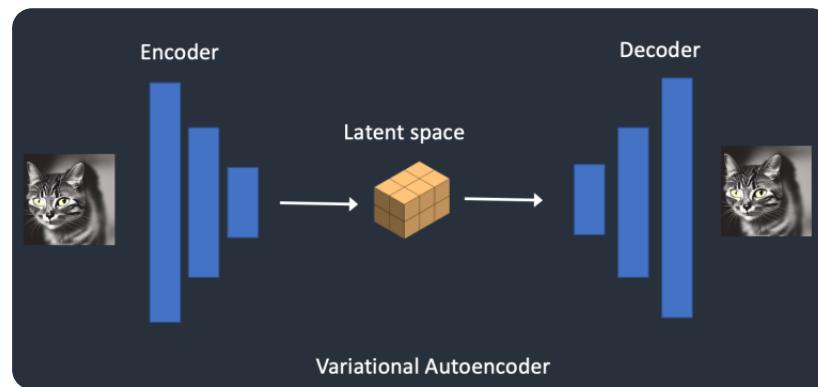
Stable Diffusion is designed to solve the speed problem. Here's how.

Stable Diffusion is a latent diffusion model. Instead of operating in the high-dimensional image space, it first compresses the image into the **latent space**. The latent space is 48 times smaller so it reaps the benefit of crunching a lot fewer numbers. That's why it's a lot faster.

Variational Autoencoder

It is done using a technique called the **variational autoencoder**. Yes, that's precisely what the VAE files are, but I will make it crystal clear later.

The Variational Autoencoder (VAE) neural network has two parts: (1) an encoder and (2) a decoder. The encoder compresses an image to a lower dimensional representation in the latent space. The decoder restores the image from the latent space.



Variational autoencoder transforms the image to and from the latent space.

The latent space of Stable Diffusion model is 4x64x64, 48 times smaller than the image pixel space. All the forward and reverse diffusions we talked about are actually done in the latent space.

So during training, **instead of generating a noisy image, it generates a random tensor in latent space** (latent noise). Instead of corrupting an image with noise, it corrupts the representation of the image in latent space with the latent noise. **The reason for doing that is it is a lot faster since the latent space is smaller.**

Image resolution

The image resolution is reflected in the size of the latent image tensor. The size of the latent image is 4x64x64 for 512x512 images only. It is 4x96x64 for a 768x512 portrait image. That's why it takes longer and more VRAM to generate a larger image.

Since Stable Diffusion v1 is fine-tuned on 512x512 images, generating images larger than 512x512 could result in duplicate objects, e.g., the [infamous two heads](https://stable-diffusion-art.com/common-problems-in-ai-images-and-how-to-fix-them/) <<https://stable-diffusion-art.com/common-problems-in-ai-images-and-how-to-fix-them/>>. If you must, keep at least one side to 512 pixels and use an [AI upscaler](https://stable-diffusion-art.com/ai-upscaler/) <<https://stable-diffusion-art.com/ai-upscaler/>> for higher resolution.

Why is latent space possible?

You may wonder why the VAE can compress an image into a much smaller latent space without losing information. The reason is, unsurprisingly, natural images are not random. They have high regularity: A face follows a specific spatial relationship between the eyes, nose, cheek, and mouth. A dog has 4 legs and is a particular shape.

In other words, the high dimensionality of images is artifactual. Natural images can be readily compressed into the much smaller latent space without losing any information. This is called the [manifold hypothesis](https://en.wikipedia.org/wiki/Manifold_hypothesis) <https://en.wikipedia.org/wiki/Manifold_hypothesis> in machine learning.

Reverse diffusion in latent space

Here's how latent reverse diffusion in Stable Diffusion works.

1. A random latent space matrix is generated.
2. The **noise predictor** estimates the noise of the latent matrix.
3. The estimated noise is then subtracted from the latent matrix.
4. Steps 2 and 3 are repeated up to specific sampling steps.
5. The **decoder** of VAE converts the latent matrix to the final image.

What is a VAE file?

[VAE files](https://stable-diffusion-art.com/how-to-use-vae/) <<https://stable-diffusion-art.com/how-to-use-vae/>> are used in Stable Diffusion v1 to improve eyes and faces. They are the **decoder of the autoencoder** we just talked about. By further fine-tuning the decoder, the model can paint finer details.

You may realize what I have mentioned previously is not entirely true. Compressing an image into the latent space *does* lose information since the original VAE did not recover the fine details. Instead, the VAE decoder is responsible for painting fine details.

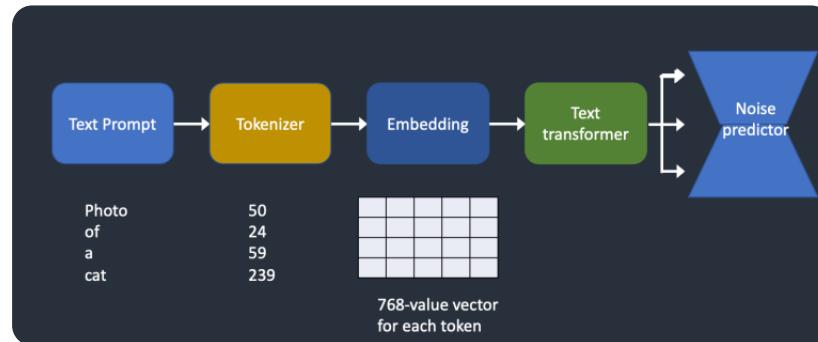
Conditioning

Our understanding is incomplete: Where does the text prompt enter the picture? Without it, Stable Diffusion is not a text-to-image model. You will either get an image of a cat or a dog without any way to control it.

This is where **conditioning** comes in. The purpose of conditioning is to **steer the noise predictor so that the predicted noise will give us what we want** after subtracting from the image.

Text conditioning (text-to-image)

Below is an overview of how a **text prompt** is processed and fed into the noise predictor. **Tokenizer** first converts each word in the prompt to a number called a **token**. Each token is then converted to a 768-value vector called **embedding**. (Yes, this is the same [embedding](https://stable-diffusion-art.com/embedding/) you used in AUTOMATIC1111) The embeddings are then processed by the **text transformer** and are ready to be consumed by the noise predictor.

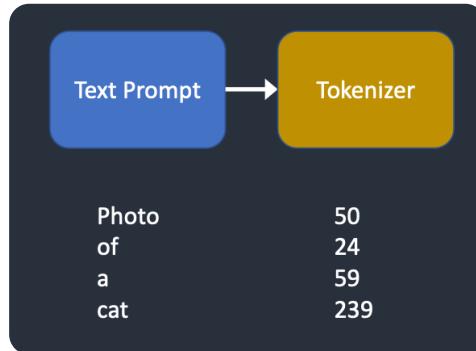


How the text prompt is processed and fed into the noise predictor to steer image generation.

Now let's look closer into each part. You can [skip to the next section](#) if the above high-level overview is good enough for you.

Check tokens and embeddings of any prompt with [this notebook](https://colab.research.google.com/github/sagiddev/stablediffusion_webui/blob/master/Stable_Diffusion_tokenizer_and_embedding_SDA.ipynb).

Tokenizer



Tokenizer.

The text prompt is first **tokenized** by a CLIP tokenizer <https://huggingface.co/docs/transformers/model_doc/clip> . CLIP is a deep learning model developed by Open AI to produce text descriptions of any images. Stable Diffusion v1 uses CLIP's tokenizer.

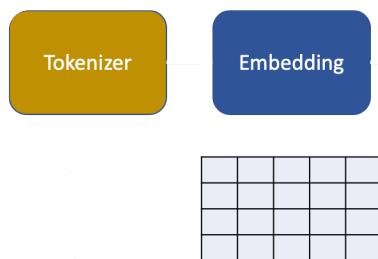
Tokenization is the computer's way of understanding words. We humans can read words, but computers can only read numbers. That's why words in a text prompt are first converted to numbers.

A tokenizer can only tokenize words it has seen during training. For example, there are "dream" and "beach" in the CLIP model but not "dreambeach". Tokenizer would break up the word "dreambeach" into two tokens "dream" and "beach". So **one word does not always mean one token!**

Another fine print is the space character is also part of a token. In the above case, the phrase "dream beach" produces two tokens "dream" and "[space]beach". These tokens are not the same as that produced by "dreambeach" which is "dream" and "beach" (without space before beach).

Stable Diffusion model is limited to using 75 tokens in a prompt. (Now you know it is not the same as 75 words!)

Embedding



Embedding.

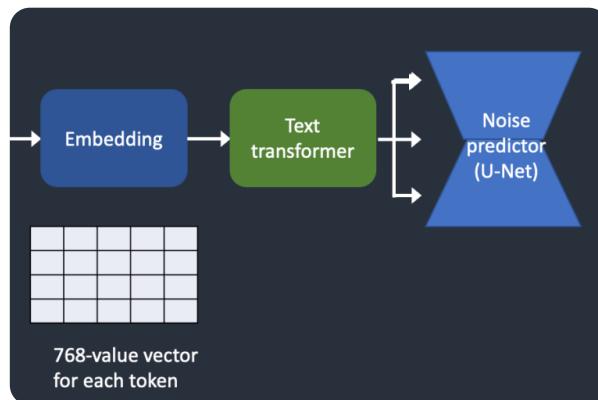
Stable diffusion v1 uses Open AI's **ViT-L/14** <

<https://github.com/CompVis/stable-diffusion>> Clip model. Embedding is a 768-value vector. Each token has its own unique embedding vector. Embedding is fixed by the CLIP model, which is learned during training.

Why do we need embedding? It's because some words are closely related to each other. We want to take advantage of this information. For example, the embeddings of *man*, *gentleman*, and *guy* are nearly identical because they can be used interchangeably. Monet, Manet, and Degas all painted in impressionist styles but in different ways. The names have close but not identical embeddings.

This is the same embedding < <https://stable-diffusion-art.com/embedding/>> we discussed for triggering a style with a keyword. Embeddings can do magic. Scientists have shown that finding the proper embeddings can trigger arbitrary objects and styles, a fine-tuning technique called **textual inversion** < <https://textual-inversion.github.io/>> .

Feeding embeddings to noise predictor



From embeddings to the noise predictor.

The embedding needs to be further processed by the **text transformer** before feeding into the noise predictor. The transformer is like a universal adapter for conditioning. In this case, its input is text embedding vectors, but it could as well be something else like class labels, images, and **depth maps** < <https://stable-diffusion-art.com/depth-to-image/>> . The transformer not only further processes the data but also **provides a mechanism to include different conditioning modalities**.

Cross-attention

The output of the text transformer is used **multiple times** by the noise predictor throughout the U-Net. The U-Net consumes it by a **cross-attention mechanism**. That's where the prompt meets the image.

Let's use the prompt "A man with blue eyes" as an example. Stable Diffusion pairs the two words "blue" and "eyes" together (self-attention within the

prompt) so that it generates a man with blue eyes but not a man with a blue shirt. It then uses this information to steer the reverse diffuse towards images containing blue eyes. (cross-attention between the prompt and the image)

A side note: Hypernetwork, a technique to fine-tune Stable Diffusion models, hijacks the cross-attention network to insert styles. LoRA models <<https://stable-diffusion-art.com/lora/>> modify the weights of the cross-attention module to change styles. The fact that modifying this module alone can fine-tune a Stable Diffusion model tells you how important this module is.

Other conditionings

The text prompt is not the only way a Stable Diffusion model can be conditioned.

Both a text prompt and a depth image are used to condition the depth-to-image <<https://stable-diffusion-art.com/depth-to-image/>> model.

ControlNet <<https://stable-diffusion-art.com/controlnet/>> conditions the noise predictor with detected outlines, human poses, etc, and achieves excellent controls over image generations.

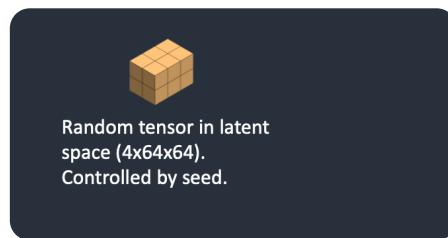
Stable Diffusion step-by-step

Now you know all the inner mechanics of Stable Diffusion, let's go through some examples of what happens under the hood.

Text-to-image

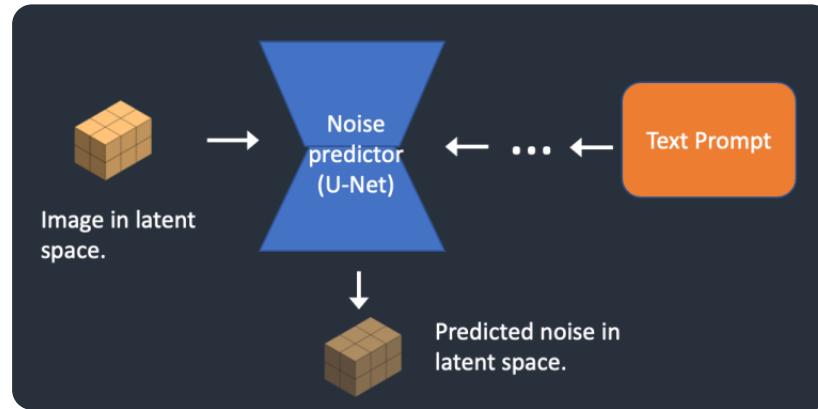
In text-to-image, you give Stable Diffusion a text prompt, and it returns an image.

Step 1. Stable Diffusion generates a random tensor in the latent space. You control this tensor by setting the seed <<https://stable-diffusion-art.com/know-these-important-parameters-for-stunning-ai-images/#Seed>> of the random number generator. If you set the seed to a certain value, you will always get the same random tensor. **This is your image in latent space.** But it is all noise for now.

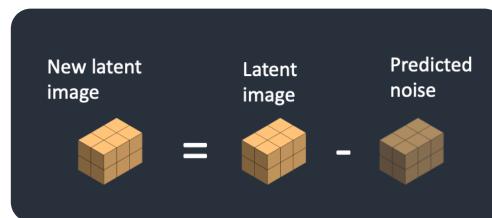


A random tensor is generated in latent space.

Step 2. The noise predictor U-Net takes the latent noisy image and text prompt as input and **predicts** the noise, also in latent space (a 4x64x64 tensor).

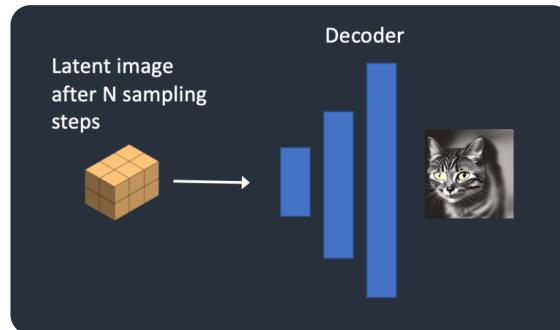


Step 3. Subtract the latent noise from the latent image. This becomes your **new latent image**.



Steps 2 and 3 are repeated for a certain number of sampling steps, for example, 20 times.

Step 4. Finally, the decoder of VAE converts the latent image back to pixel space. This is the image you get after running Stable Diffusion.



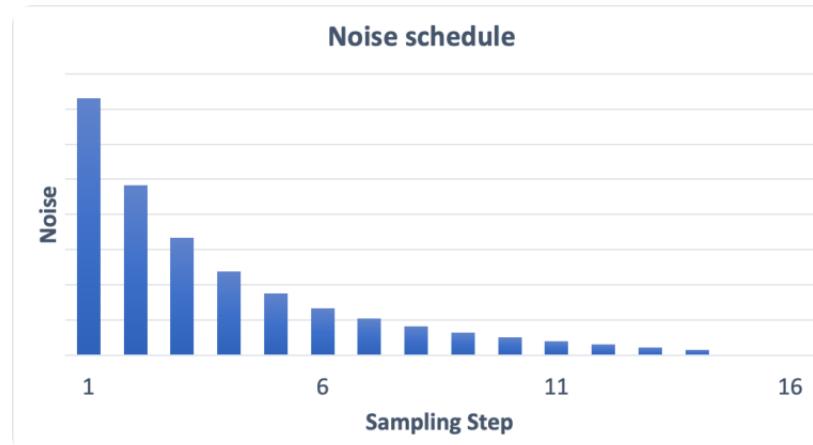
Here's how the image evolves in each sampling step.



Image at each sampling step.

Noise schedule

The image changes from noisy to clean. Do you wonder if the noise predictor not working well in the initial steps? Actually, this is only partly true. The real reason is we try to get to an **expected noise** at each sampling step. This is called the **noise schedule**. Below is an example.



A noise schedule for 15 sampling steps.

The noise schedule is something we define. We can choose to subtract the same amount of noise at each step. Or we can subtract more in the beginning, like above. The sampler subtracts just enough noise in each step to reach the expected noise in the next step. That's what you see in the step-by-step image.

Image-to-image

Image-to-image is a method first proposed in the SDEdit < <https://arxiv.org/abs/2108.01073>> method. SDEdit can be applied to any diffusion model. So we have image-to-image for Stable Diffusion (a latent diffusion model).

An input image and a text prompt are supplied as the input in image-to-image. The generated image will be conditioned by both the input image and text prompt. for example, using this amateur drawing and the prompt "photo of perfect green apple with stem, water droplets, dramatic lighting" as inputs, image-to-image can turn it into a professional drawing:

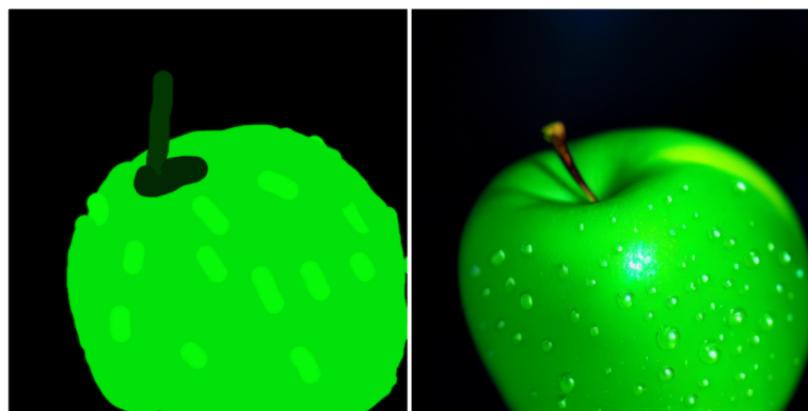
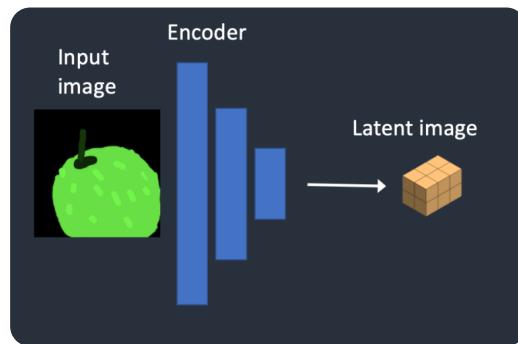


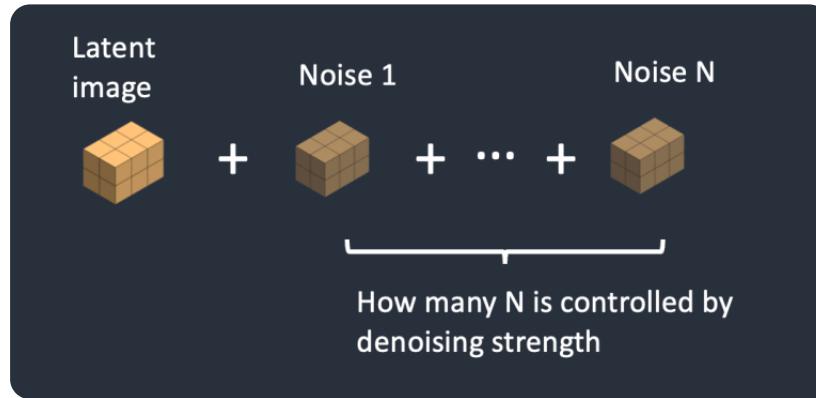
Image-to-image

Now here's the step-by-step process.

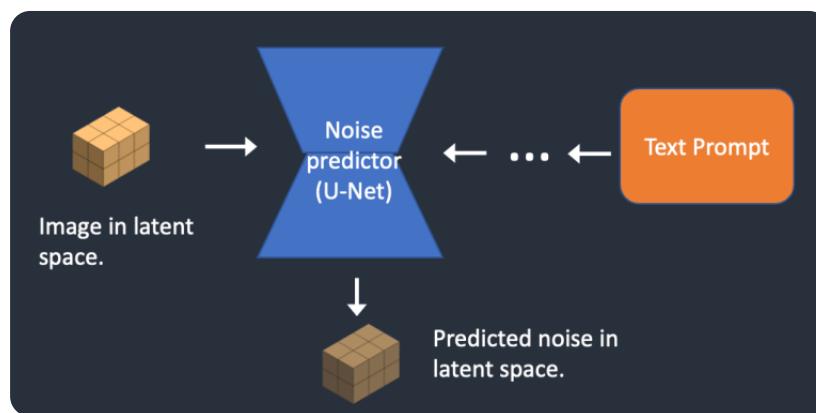
Step 1. The input image is encoded to latent space.



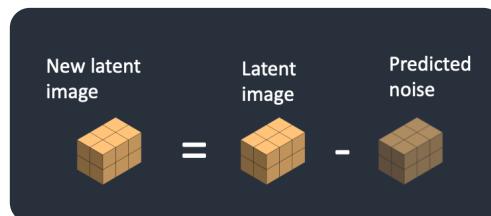
Step 2. Noise is added to the latent image. Denoising strength < https://stable-diffusion-art.com/inpainting_basics/#Denoising_strength> controls how much noise is added. If it is 0, no noise is added. If it is 1, the maximum amount of noise is added so that the latent image becomes a complete random tensor.



Step 3. The noise predictor U-Net takes the latent noisy image and text prompt as input and predicts the noise in latent space (a 4x64x64 tensor).

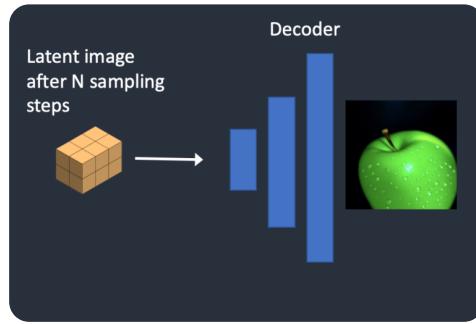


Step 4. Subtract the latent noise from the latent image. This becomes your new latent image.



Steps 3 and 4 are repeated for a certain number of sampling steps, for example, 20 times.

Step 5. Finally, the decoder of VAE converts the latent image back to pixel space. This is the image you get after running image-to-image.



So now you know what image-to-image is: All it does is to set the initial latent image with a bit of noise and a bit of input image. Setting denoising strength to 1 is equivalent to text-to-image because the initial latent image is entirely random noise.

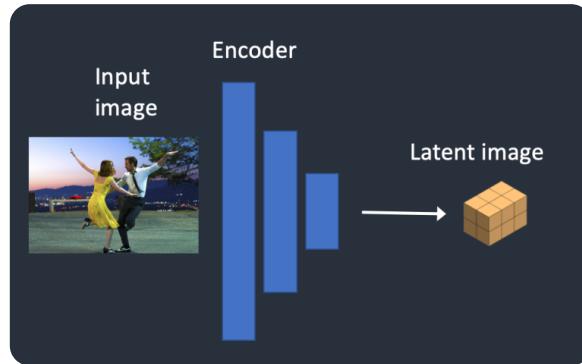
Inpainting

Inpainting is really just a particular case of image-to-image. Noise is added to the parts of the image you wanted to inpaint. The amount of noise is similarly controlled by denoising strength.

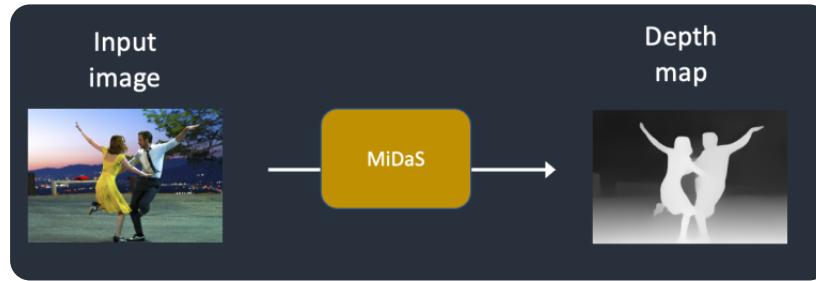
Depth-to-image

Depth-to-image < <https://stable-diffusion-art.com/depth-to-image/>> is an enhancement to image-to-image; it generates new images with additional conditioning using a depth map.

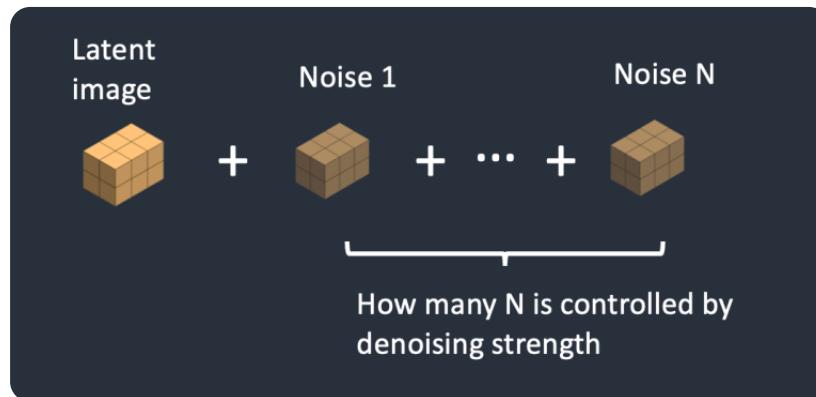
Step 1. The input image is encoded into the latent state



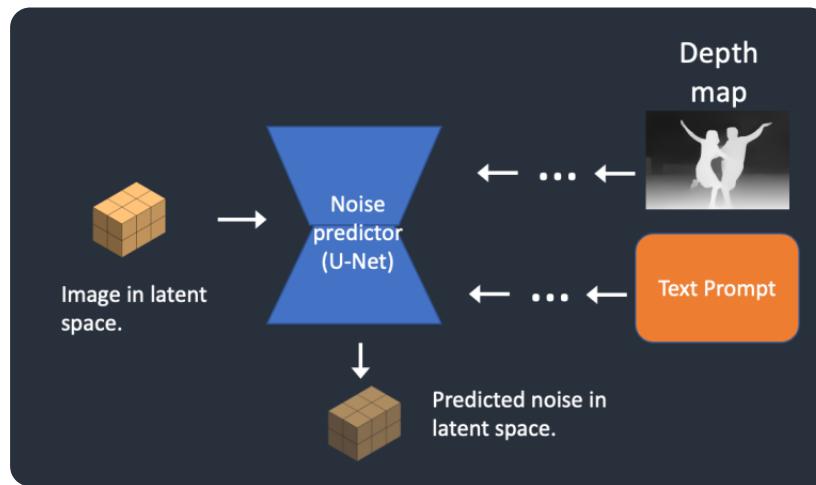
Step 2. MiDaS < <https://github.com/isl-org/MiDaS>> (an AI depth model) estimates the depth map from the input image.



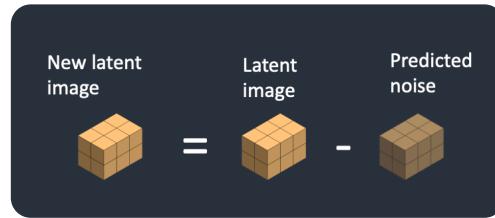
Step 3. Noise is added to the latent image. Denoising strength controls how much noise is added. If the denoising strength is 0, no noise is added. If the denoising strength is 1, the maximum noise is added so that the latent image becomes a random tensor.



Step 4. The noise predictor estimates the noise of the latent space, **conditioned by the text prompt and the depth map**.

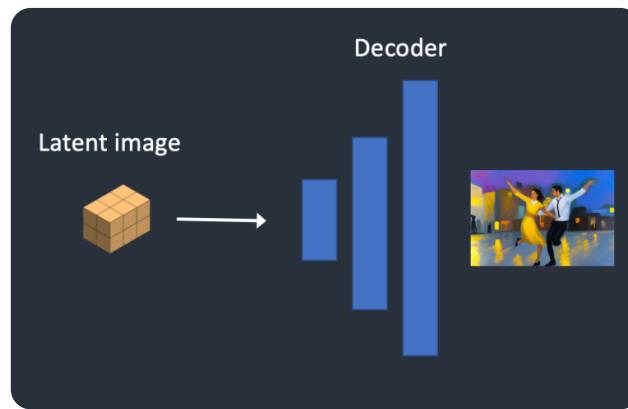


Step 5. Subtract the latent noise from the latent image. This becomes your **new latent image**.



Steps 4 and 5 are repeated for the number of sampling steps.

Step 6. The decoder of VAE decodes the latent image. Now you get the final image from depth-to-image.



What is CFG value?

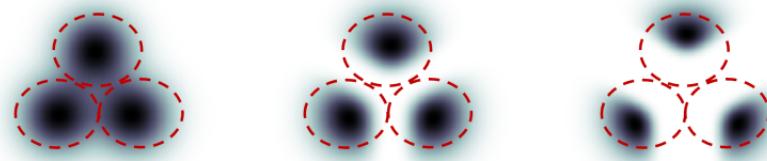
This write-up won't be complete without explaining Classifier-Free Guidance (CFG), a value AI artists tinker with every day. To understand what it is, we will need to first touch on its predecessor, **classifier guidance...**

Classifier guidance

Classifier guidance < <https://arxiv.org/abs/2105.05233>> is a way to incorporate **image labels** in diffusion models. You can use a label to guide the diffusion process. For example, the label "cat" steers the reverse diffusion process to generate photos of cats.

The **classifier guidance scale** is a parameter for controlling how closely should the diffusion process follow the label.

Below is an example I stole from this paper <<https://arxiv.org/abs/2207.12598>>. Suppose there are 3 groups of images with labels "cat", "dog", and "human". If the diffusion is unguided, the model will draw samples from each group's total population, but sometimes it may draw images that could fit two labels, e.g. a boy petting a dog.



Classifier guidance. Left: unguided. Middle: small guidance scale. Right: large guidance scale.

With high classifier guidance, the images produced by the diffusion model would be biased toward the **extreme or unambiguous examples**. If you ask the model for a cat, it will return an image that is unambiguously a cat and nothing else.

The **classifier guidance scale** controls how closely the guidance is followed. In the figure above, the sampling on the right has a higher classifier guidance scale than the one in the middle. In practice, this scale value is simply the multiplier to the drift term toward the data with that label.

Classifier-free guidance

Although classifier guidance achieved record-breaking performance, it needs an extra model to provide that guidance. This has presented some difficulties in training.

Classifier-free guidance < <https://arxiv.org/abs/2207.12598> >, in its authors' terms, is a way to achieve "classifier guidance without a classifier". Instead of using class labels and a separate model for guidance, they proposed to use image captions and train a **conditional diffusion model**, exactly like the one we **discussed** in text-to-image.

They put the classifier part as **conditioning of the noise predictor U-Net**, achieving the so-called "classifier-free" (i.e. without a separate image classifier) guidance in image generation.

The text prompt provides this guidance in text-to-image.

CFG value

Now we have a classifier-free diffusion process via conditioning, how do we control how much the guidance should be followed?

Classifier-free guidance (CFG) scale is a value that controls how much the text prompt conditions the diffusion process. The image generation is **unconditioned** (i.e. the prompt is ignored) when it is set to 0. A higher value steers the diffusion towards the prompt.

Stable Diffusion v1 vs v2

This is already a long post, but it won't be complete without comparing the difference between v1 and v2 <<https://stable-diffusion-art.com/how-to-run-stable-diffusion-2-0/>> models.

Model difference

Stable Diffusion v2 uses OpenClip <<https://stability.ai/blog/stable-diffusion-v2-release>> for text embedding. Stable Diffusion v1 uses Open AI's CLIP ViT-L/14 <<https://github.com/CompVis/stable-diffusion>> for text embedding. The reasons for this change are

- OpenClip is up five times larger. A larger text encoder model improves image quality.
- Although Open AI's CLIP models are open-source, the models were trained with proprietary data. Switching to the OpenClip model gives researchers more transparency in studying and optimizing the model. It is better for long-term development.

Training data difference

Stable Diffusion v1.4 is trained <<https://huggingface.co/CompVis>> with

- 237k steps at resolution 256x256 on laion2B-en <<https://huggingface.co/datasets/laion/laion2B-en>> dataset.
- 194k steps at resolution 512x512 on laion-high-resolution <<https://huggingface.co/datasets/laion/laion-high-resolution>> .
- 225k steps at 512x512 on "laion-aesthetics v2 5+ <<https://laion.ai/blog/laion-aesthetics/>> ", with 10% dropping of text conditioning.

Stable Diffusion v2 is trained with <<https://huggingface.co/stabilityai/stable-diffusion-2-base>>

- 550k steps at the resolution 256x256 on a subset of LAION-5B <<https://laion.ai/blog/laion-5b/>> filtered for explicit pornographic material, using the LAION-NSFW classifier <<https://github.com/LAION-AI/CLIP-based-NSFW-Detector>> with punsafe=0.1 and an aesthetic score <<https://github.com/christophschuhmann/improved-aesthetic-predictor>> >= 4.5 .
- 850k steps at the resolution 512x512 on the same dataset on images with resolution >= 512x512 .
- 150k steps using a v-objective <<https://arxiv.org/abs/2202.00512>> on the same dataset.
- Resumed for another 140k steps on 768x768 images.

Stable Diffusion v2.1 <<https://huggingface.co/stabilityai/stable-diffusion-2-1>> is fine-tuned on v2.0

- additional 55k steps on the same dataset (with punsafe=0.1)
- another 155k extra steps with punsafe=0.98

So basically, they **turned off the NSFW filter in the last training steps**.

Outcome difference

Users generally find it harder to use Stable Diffusion v2 to control styles and generate celebrities. Although Stability AI did not explicitly filter out artist and celebrity names, their effects are much weaker in v2. This is likely due to the difference in training data. Open AI's proprietary data may have more artwork and celebrity photos. Their data is probably highly filtered so that everything and everyone looks fine and pretty.

Some interesting reads

- **Stable Diffusion v1.4 press release** < <https://stability.ai/blog/stable-diffusion-public-release>>
- **Stable Diffusion v2 press release** < <https://stability.ai/blog/stable-diffusion-v2-release>>
- **Stable Diffusion v2.1 press release** < <https://stability.ai/blog/stablediffusion2-1-release7-dec-2022>>
- **High-Resolution Image Synthesis with Latent Diffusion Models** < <https://arxiv.org/abs/2112.10752>> – research paper introducing Stable Diffusion
- **The Illustrated Stable Diffusion** < <https://jalammar.github.io/illustrated-stable-diffusion/>> – Some good details in model architecture
- **Stable Diffusion 2** < <https://huggingface.co/stabilityai/stable-diffusion-2>> – Official model page
- **Diffusion Models Beat GANs on Image Synthesis** < <https://arxiv.org/abs/2105.05233>> – Research paper introducing classifier guidance
- **Classifier-Free Diffusion Guidance** < <https://arxiv.org/abs/2207.12598>> – Research paper introducing classifier-free guidance
- **Deep Unsupervised Learning using Nonequilibrium Thermodynamics** < <https://arxiv.org/abs/1503.03585>> – Reverse diffusion process

YOU MAY BE INTERESTED IN

- Prompt Generator:** A step-by-step system to build high-quality prompts. < https://andrewongai.gumroad.com/l/stable_diffusion_prompt_generator>
- Beginner's Guide Book :** All you need to be an expert user. < <https://andrewongai.gumroad.com/l/beginners-guide-to-stable-diffusion-ebook>>



< <https://www.buymeacoffee.com/sdart>>

< <https://www.addtoany.com/share#url=https%3A%2F%2Fstable-diffusion-art.com%2Fhow-stable-diffusion-work%2F&title=How%20does%20Stable%20Diffusi>

21 comments



Ian

July 9, 2023 at 12:04 pm

I missed the part whereby it goes from an amateur drawing of a green apple to a photorealistic green apple. Where does it get the information used to generate a photorealistic apple? Does it uses preexisting photorealistic apples and see which one closely matches with all the properties of the latent amateur apple?

Thanks,

Ian



Andrew < <https://stable-diffusion-art.com>>

July 11, 2023 at 10:44 pm

The model was trained to generate images matching the text prompt. It is not by storing the images but learning the distribution of what a real image should look like.

In img2img, it takes the amateur picture as initial condition and generate a professional drawing of an apple (as specified in the prompt) to generate one matches closely.



Jamie

July 2, 2023 at 8:03 am

Hey Andrew, is this simplified explanation of the noise diffusion process true?

Theoretically, it's like inserting an 'ice cream' mosaic with hundreds of other tesserae (rectangular slabs used to create a mosaic) and then asking a highly intelligent artist to watch them being removed to restore the original image. During this process, the artist learns how to understand and reinterpret the 'ice cream' image in other mosaics. The artist is trained to do this with millions of other images in mosaics so that they can create entirely new ones determined by the requests (or text prompts) of the person commissioning them.



Andrew < <https://stable-diffusion-art.com>>

July 3, 2023 at 10:01 pm

This could be an analogy but not entirely correct. Over time, the artist learned how the images supposed to look like with certain descriptions, so she draws statistically likely ones.



InklingSutra < <http://inklingsutra.wordpress.com>>

June 13, 2023 at 4:10 am

these insights are way better than anything available out there on other sites.
Thank @andrew!



Linh

June 3, 2023 at 2:52 am

This blog is absolutely fantastic! I am thoroughly impressed with the way you incorporate captivating images and diagrams. They are incredibly effective and have been immensely helpful to me. I am sincerely grateful for your exceptional work. Thank you so much!



Lei

June 1, 2023 at 8:15 am

Great article,
Thank you very much for the clear and concise introduction to the principles of Stable Diffusion.



Kaiwei Chen

May 18, 2023 at 5:52 am

The best blog i've ever seen. But I am still confused about one question: how the model generate images based on multi-prompt with different weights? How the different weight feature works? Is it realized by different CFG?



Andrew < <https://stable-diffusion-art.com>>

May 18, 2023 at 8:56 am

Hi good question!

The weight works as follows.

1. Each token of the prompt is converted to embedding.
2. Each embedding is multiplied by the token's weight.

In other words, it is a multiplier to the embedding of the token. It is different from the CFG scale which applies to the whole conditioning process.



Adrian < <http://gravatar.com/osssman>>

March 23, 2023 at 10:44 pm

Thanks for a great explanation, still got googled here while searching for how resolution settings work in SD (width x height), how does the model adapt to new resolution?



Andrew < <https://stable-diffusion-art.com>>

March 24, 2023 at 9:07 am

Hi, the image size is reflected in the size of the latent image tensor. Eg. the size of the latent image is 4x64x64 for a 512x512 image but is 4x96x64 for a 768x512 image. The noise predictor U-Net can consume latent states of different sizes, much like CNN for images.



vh hb (@Gned_l) < https://twitter.com/Gned_l>

March 23, 2023 at 9:20 am

Very nice article, thanks for putting together this information



Corry

March 17, 2023 at 12:26 pm

Great post, learned a lot for Stable Diffusion! Just a quick question for VAE part. You mentioned "the latent space of Stable Diffusion model is "4x64x64" which is different from [<https://x64huggingface.co/blog/stable_diffusion>](https://x64huggingface.co/blog/stable_diffusion), "(3, 64, 64) in latent space, which requires $8 \times 8 = 64$ times less memory". May I know why is different? Thanks,



Andrew < <https://stable-diffusion-art.com>>



March 17, 2023 at 12:57 pm

I remember I checked the dimension in the model when I wrote this post. My numbers are the same as another post
[https://jalammar.github.io/illustrated-stable-diffusion/ <](https://jalammar.github.io/illustrated-stable-diffusion/)
[https://jalammar.github.io/illustrated-stable-diffusion/> . But either way,](https://jalammar.github.io/illustrated-stable-diffusion/>)
this is just how the dimension is defined and it won't affect how the model
works.



Chlowden

March 11, 2023 at 5:45 am

Thank you for shedding light in the mysteries of SD. This blog is fantastic.



Thoughtship

March 2, 2023 at 8:06 am

2nd time writing this out. This is a fantastic post Andrew.
You took something that is completely foreign and hard to comprehend for
most people and explained it in a way that provides a decent high level
understanding in a single post.
Your explanations go the right depth to understand what is *actually going on*
when using Stable diffusion.

Using stable diffusion is a bit like driving a car. You don't need to know how the
engine works to drive a car. But it sure helps when you're trying to troubleshoot
or improve it. Thank you so much for this.



Andrew < <https://stable-diffusion-art.com>>

March 2, 2023 at 4:42 pm

Thanks for the comment!!



Aarshay

February 8, 2023 at 5:54 am

i browsed through various blogs and youtube videos before this, but nothing
matches the clarity of thought here. thanks a lot!



[Andrew < https://stable-diffusion-art.com >](https://stable-diffusion-art.com)

February 9, 2023 at 12:41 pm

Thank you!



Atharv Jairath

January 27, 2023 at 10:57 am

Best Blog, thanks for using diagrams



[Andrew < https://stable-diffusion-art.com >](https://stable-diffusion-art.com)

January 27, 2023 at 11:02 am

Thanks! I spent an insane amount of time on this post...

Subscribe!

Get updates on the latest tutorials, prompts, and exclusive content.

Your email

[Subscribe](#)

Welcome to **Stable Diffusion Art**! We want to make Stable Diffusion AI accessible to everyone. You will find tutorials and resources to help you use this transformative tech here.

Latest Articles



[< https://andrewongai.gumroad.com/l/stable_diffusion_prompt_generator >](https://andrewongai.gumroad.com/l/stable_diffusion_prompt_generator)

Prompt Generator for Stable Diffusion <
[< https://andrewongai.gumroad.com/l/stable_diffusion_prompt_generator >](https://andrewongai.gumroad.com/l/stable_diffusion_prompt_generator)



< [< https://andrewongai.gumroad.com/l/beginners-guide-to-stable-diffusion-ebook >](https://andrewongai.gumroad.com/l/beginners-guide-to-stable-diffusion-ebook)

Beginner's Guide eBook <
[< https://andrewongai.gumroad.com/l/beginners-guide-to-stable-diffusion-ebook >](https://andrewongai.gumroad.com/l/beginners-guide-to-stable-diffusion-ebook)

STABLE DIFFUSION ART < [HTTPS://STABLE-DIFFUSION-ART.COM/](https://stable-diffusion-art.com/) >

© Sagio Development LLC, 2023. All rights reserved.

