

Multimodal Deep Learning

Taxonomy of Generative Models

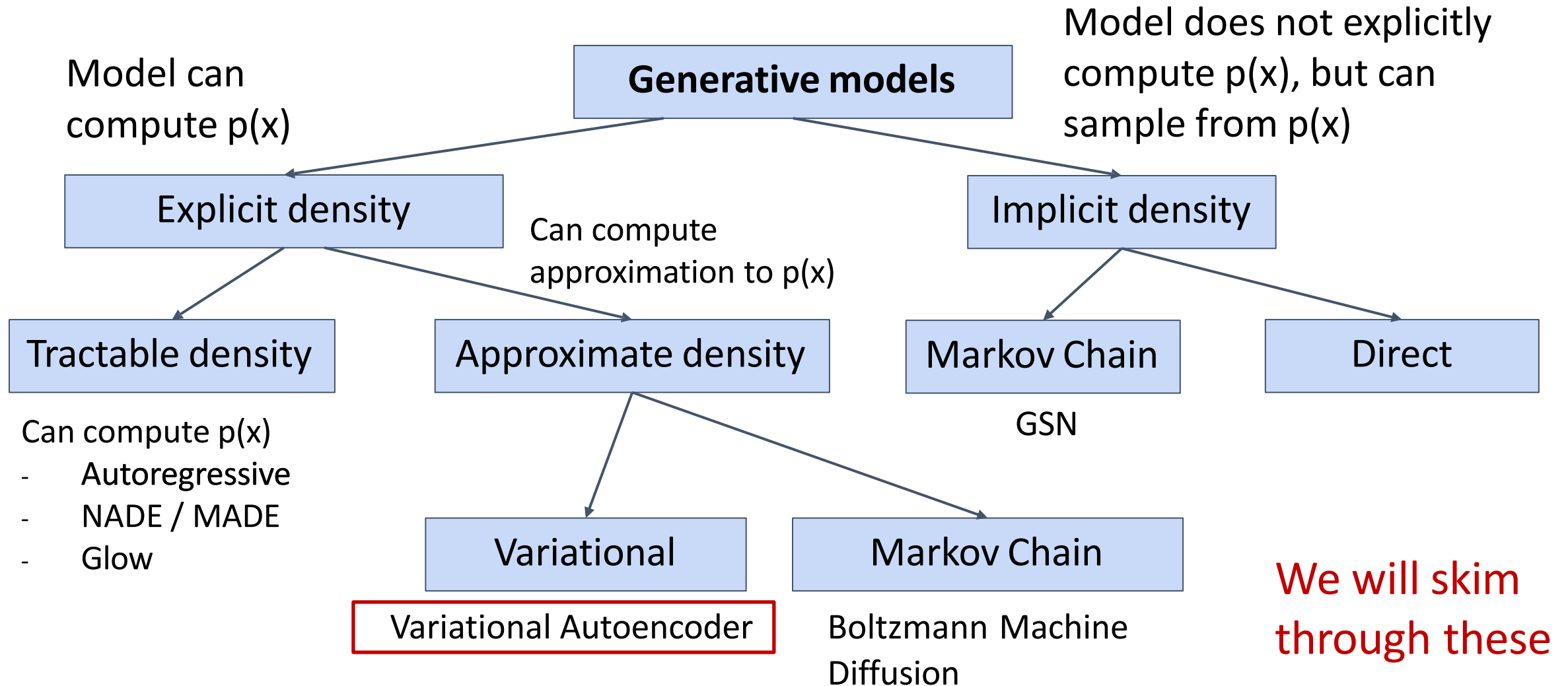
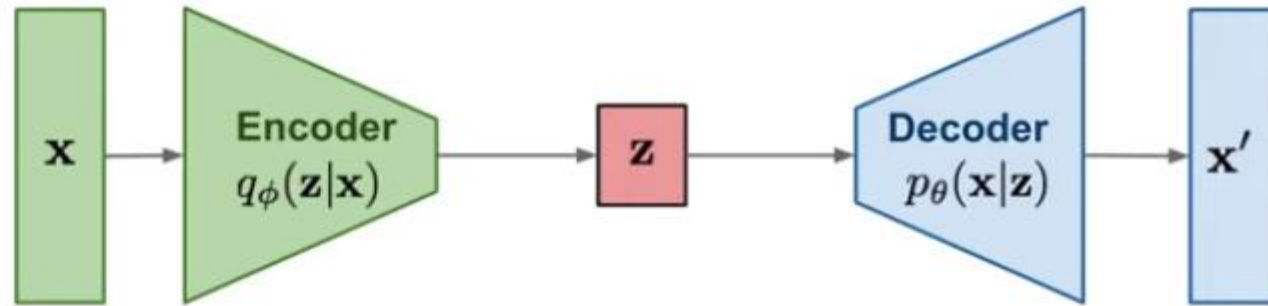


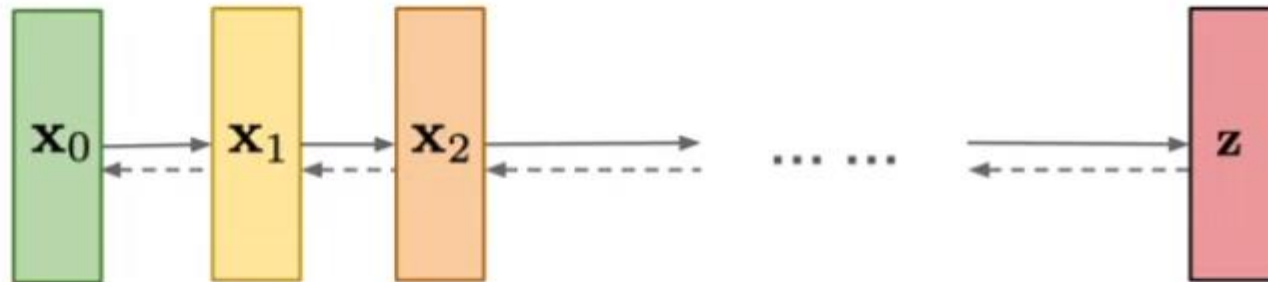
Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

VAE: maximize
variational lower bound



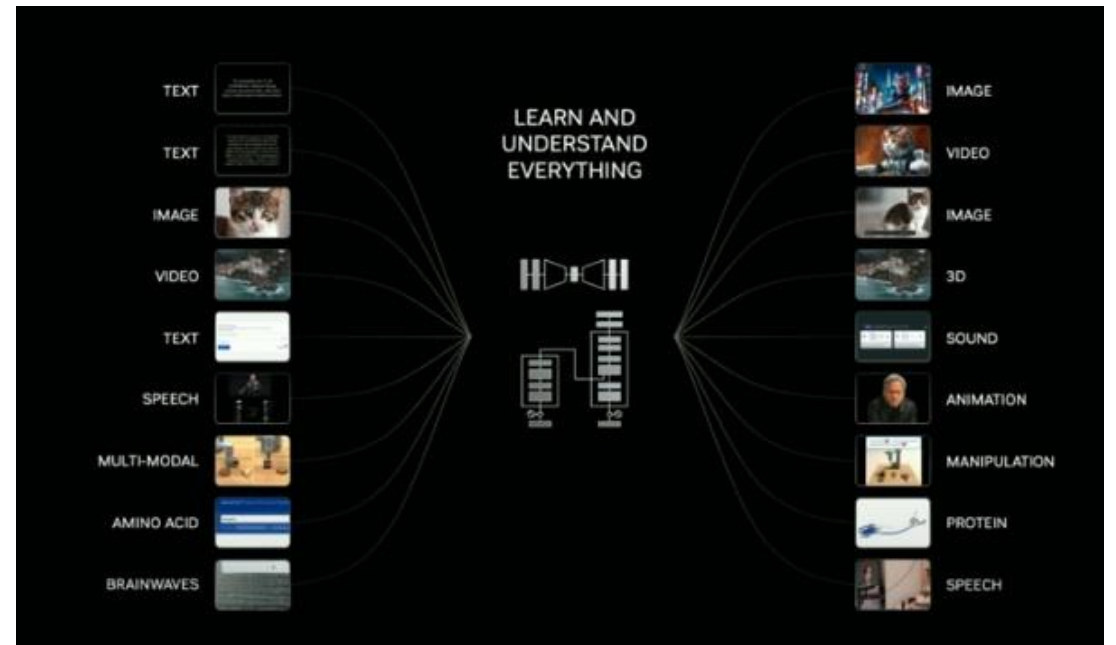
Diffusion models:
Gradually add Gaussian
noise and then reverse



Multimodal Machine Learning

Definition

- Multimodal machine learning is a type of artificial intelligence (AI) that involves training models to process and understand data from **multiple modalities** or sources, such as images, text, speech, and sensor data. The goal of multimodal machine learning is to enable machines to **learn from and make decisions based on complex and diverse data sources**, similar to how humans perceive and interact with the world.



TEXT to IMAGE

- Midjourney
- DALL-E
- Imagen
- Muse
- Stable Diffusion

...

TEXT to MUSIC

- MusicLM
- MusicGen
- Stable Audio
- Suno
- ...

TEXT to VIDEO

- Imagen Video
- Lumiere
- Emu Video
- Stable Video Diffusion
- Sora

...

Multimodal learning for Image generation

The State of Image Generation (2025 Feb)

Closed Source

-  Recraft V3
-  Imagen 3 (v002)
-  Midjourney v6.1
-  Ideogram v2 Turbo
-  Playground v3 (beta)
-  Adobe Firefly 3
-  DALI 3 HD
-  Luma Photon Flash
-  Amazon Titan G1 v2 (Standard)
-  FLUX1.1 [pro]

Open Source

-  Stable Diffusion 3.5 Large Turbo
-  SDXL-Turbo
-  Janus Pro
-  Openjourney-v4
-  Dreamlike-photoreal-2.0
-  FLUX.1 [dev]

© [2025] [The Nuanced Perspective]. All rights reserved.
Reproduction or distribution without permission is prohibited.

For permissions, collaborations, or inquiries, please
contact Aishwarya Naresh Reganti (aish.nr@gmail.com)

Stable Diffusion

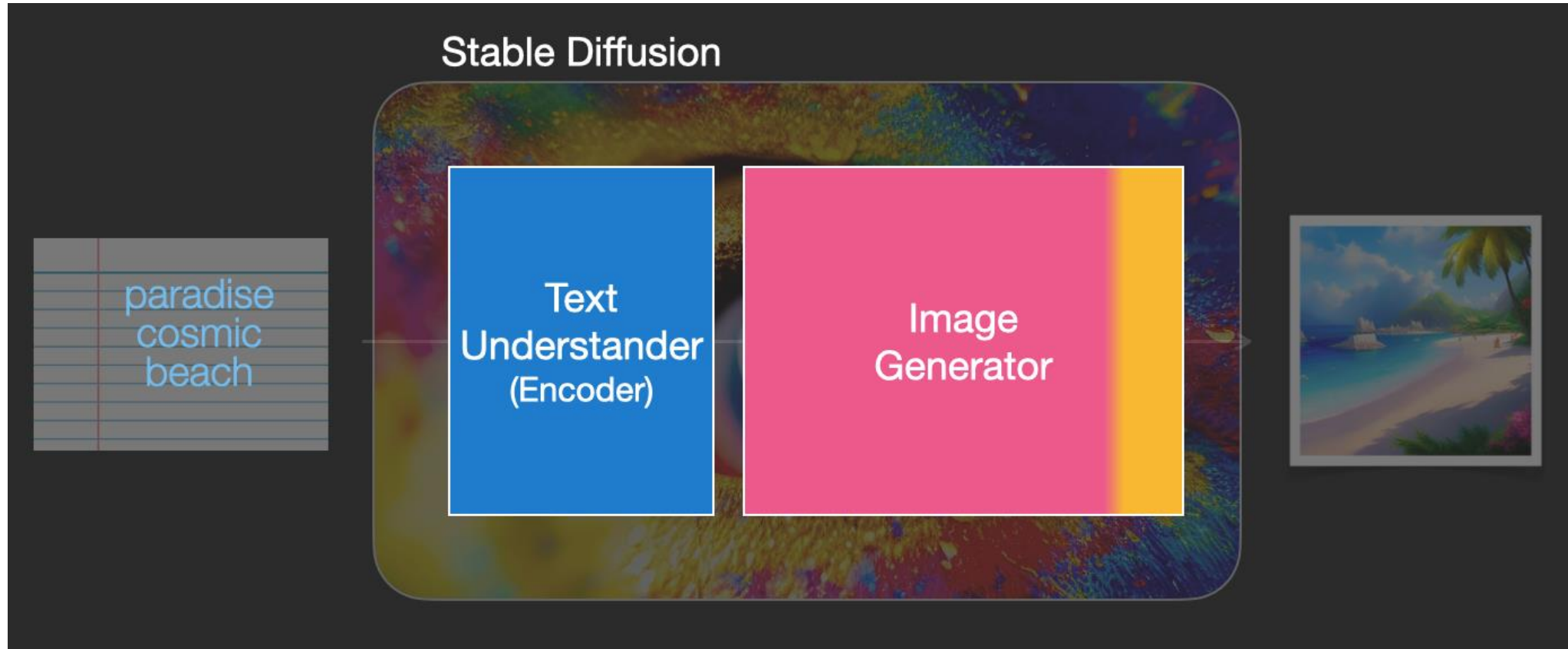


text2image



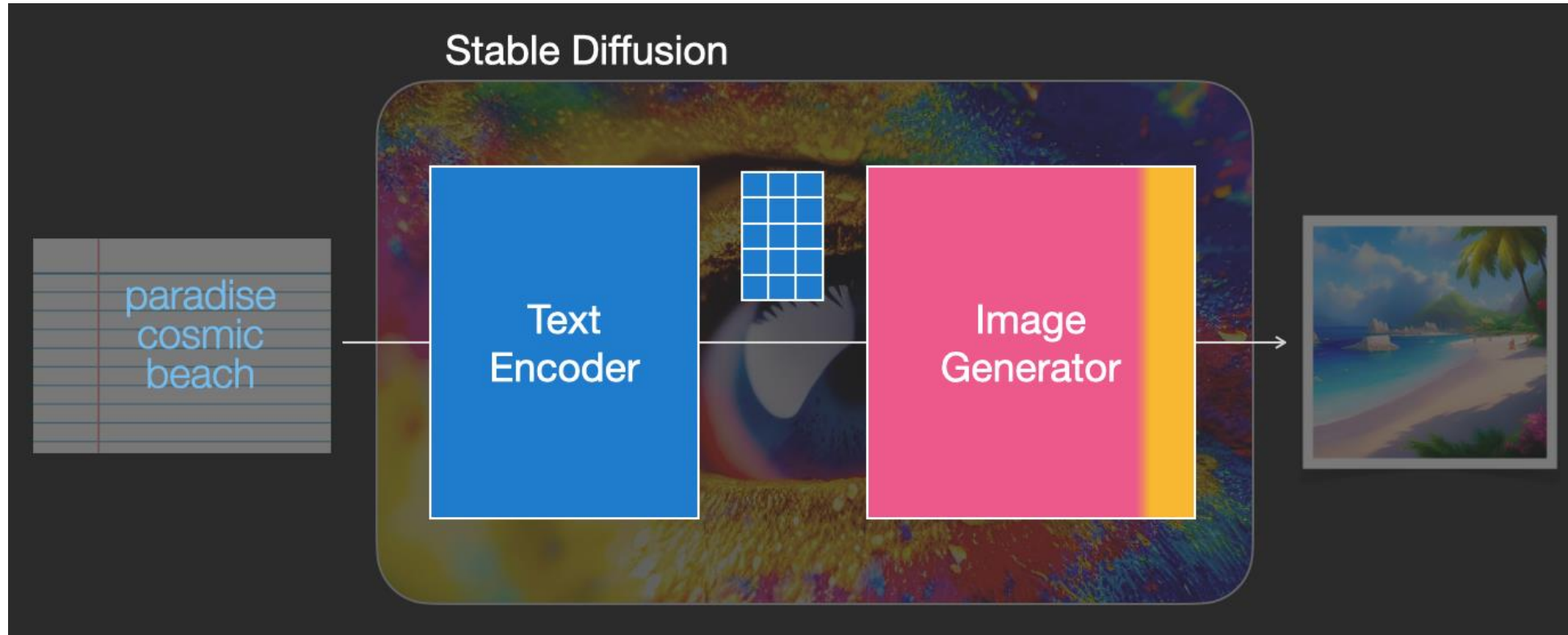
image2image

The Components of Stable Diffusion



As we look under the hood, the first observation we can make is that there's a text-understanding component that translates the text information into a numeric representation that captures the ideas in the text.

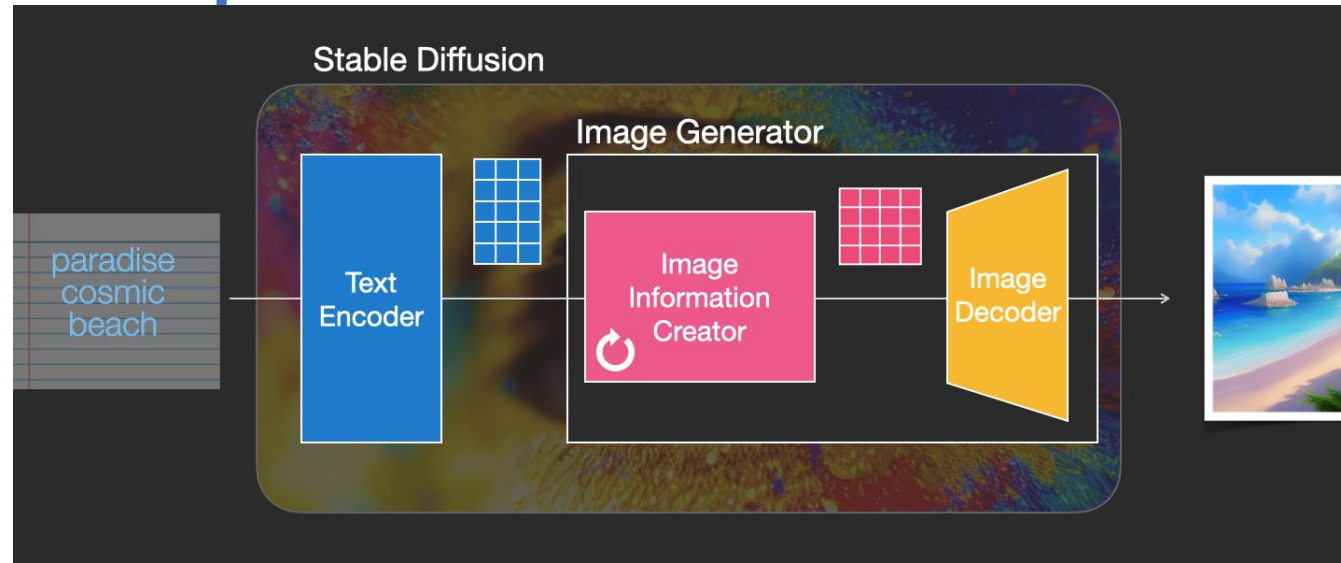
The Components of Stable Diffusion



text encoder is a special Transformer language model (technically: the text encoder of a CLIP model). It takes the input text and outputs a list of numbers representing each word/token in the text (a vector per token).

That information is then presented to the Image Generator, which is composed of a couple of components itself.

The Components of Stable Diffusion



The image generator goes through two stages:

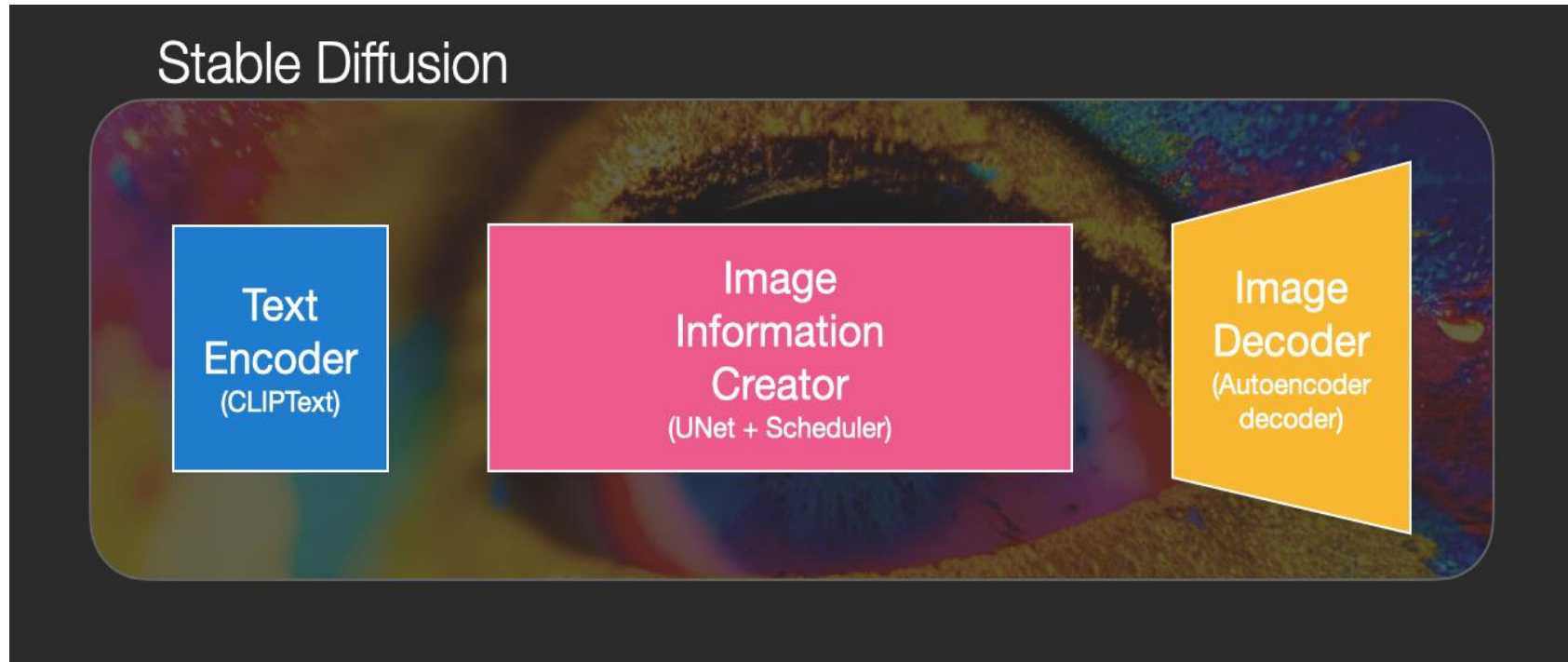
1- Image information creator

- This component is the secret sauce of Stable Diffusion. It's where a lot of the performance gain over previous models is achieved.
- This component runs for multiple steps to generate image information. This is the **steps** parameter in Stable Diffusion interfaces and libraries which often defaults to 50 or 100.
- The image information creator works completely in the *image information space* (or *latent space*). This property makes it faster than previous diffusion models that worked in pixel space. In technical terms, this component is made up of a **UNet** neural network and a **scheduling** algorithm.
- The word "diffusion" describes what happens in this component. It is the step by step processing of information that leads to a high-quality image being generated in the end (by the next component, the image decoder).

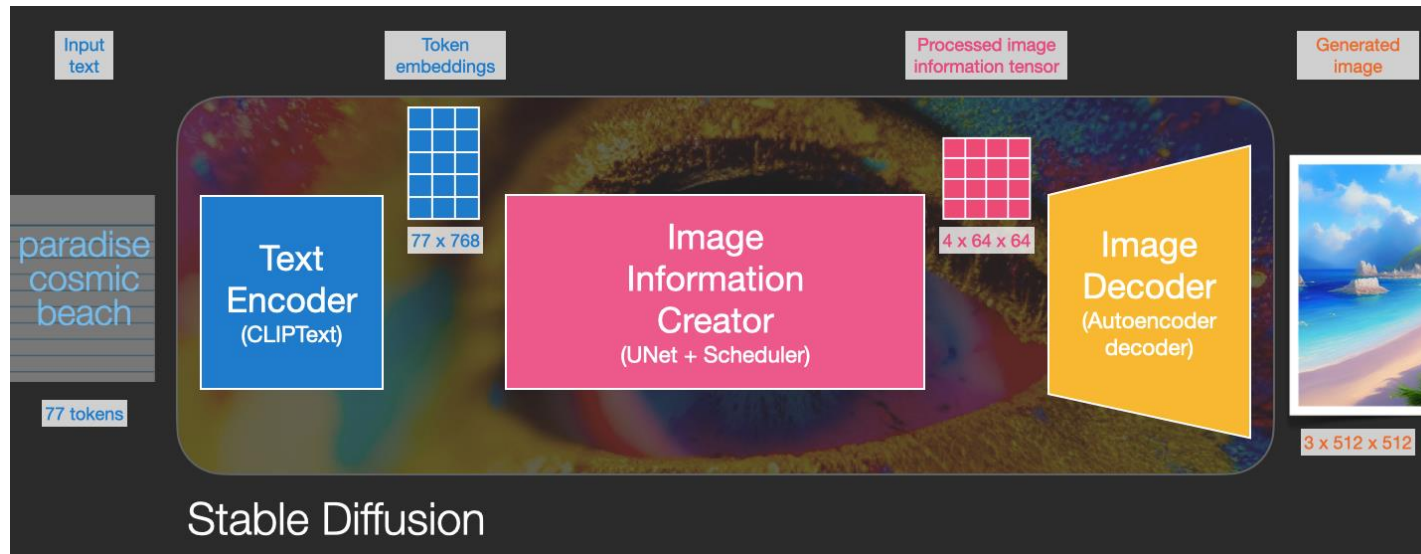
The Components of Stable Diffusion

2- Image Decoder

The image decoder paints a picture from the information it got from the information creator. It runs only once at the end of the process to produce the final pixel image.



The Components of Stable Diffusion



With this we come to see the three main components (each with its own neural network) that make up Stable Diffusion:

- **ClipText** for text encoding.

Input: text.

Output: 77 token embeddings vectors, each in 768 dimensions.(for v1)

- **UNet + Scheduler** to gradually process/diffuse information in the information (latent) space.

Input: text embeddings and a starting multi-dimensional array (structured lists of numbers, also called a *tensor*) made up of noise.

Output: A processed information array

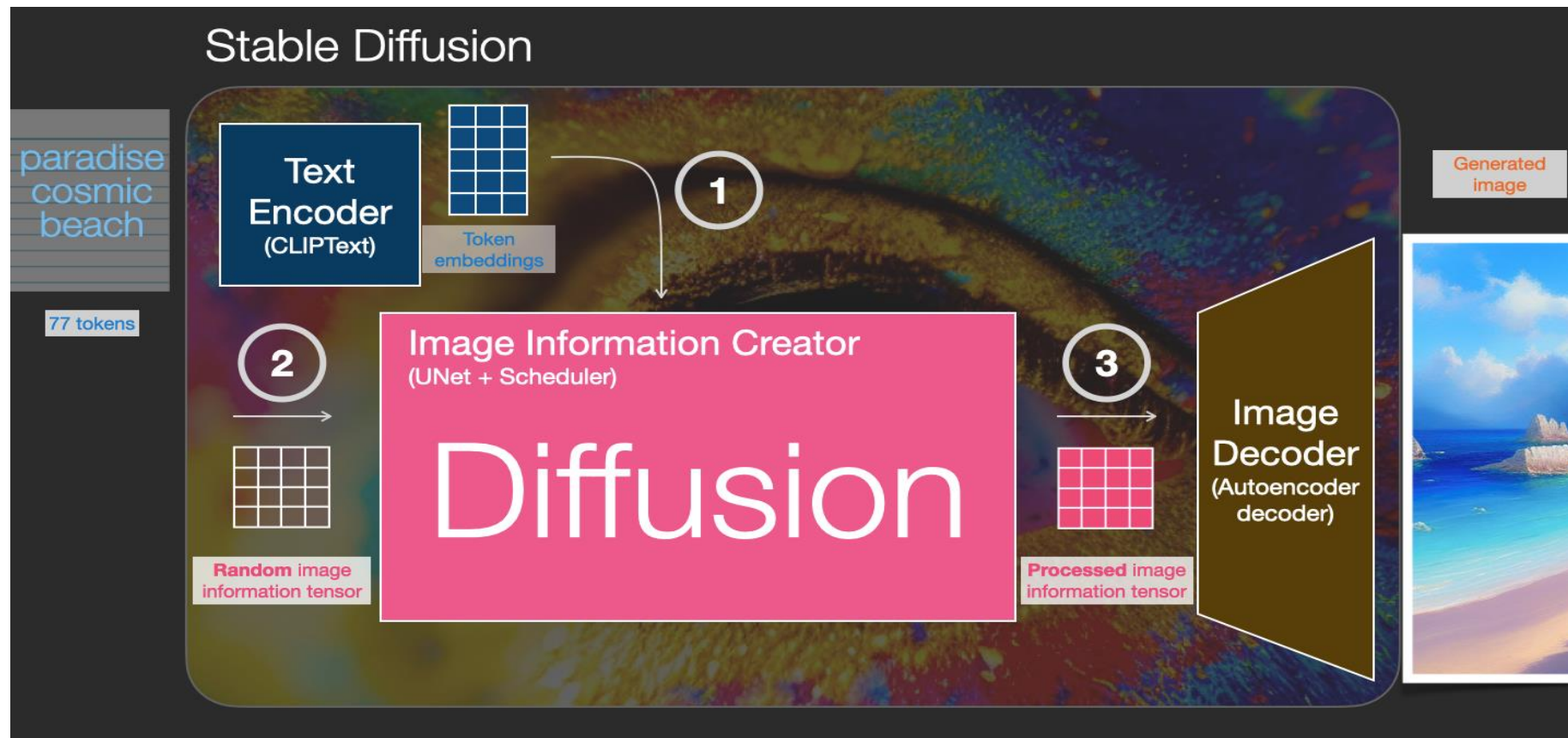
- **Autoencoder Decoder** that paints the final image using the processed information array.

Input: The processed information array (dimensions: (4,64,64))

Output: The resulting image (dimensions: (3, 512, 512) which are (red/green/blue, width, height))

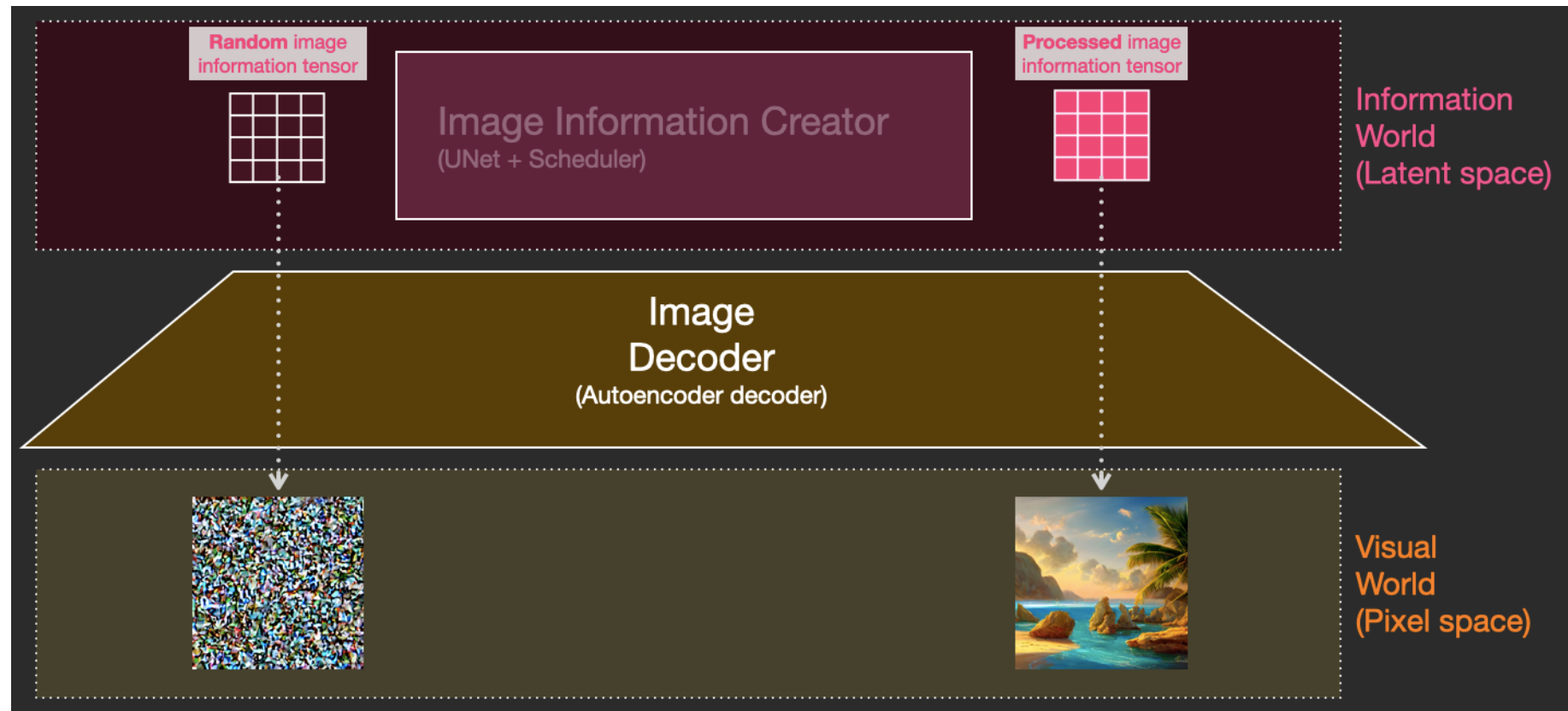
What is Diffusion Anyway?

Diffusion is the process that takes place inside the pink “image information creator” component. Having the token embeddings that represent the input text, and a random starting *image information array* (these are also called *latents*), the process produces an information array that the image decoder uses to paint the final image.



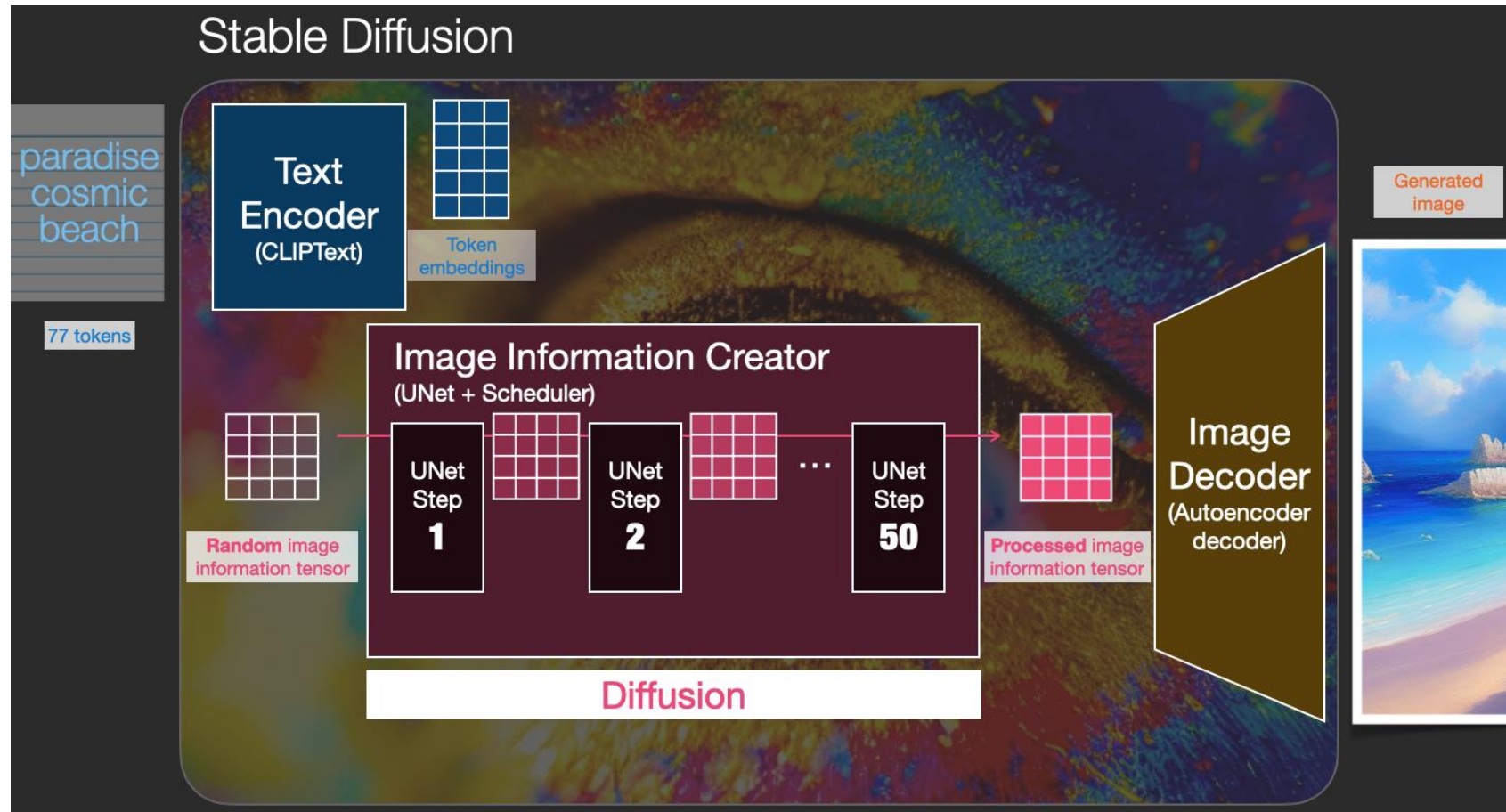
What is Diffusion Anyway?

This process happens in a step-by-step fashion. Each step adds more relevant information. To get an intuition of the process, we can inspect the random latents array, and see that it translates to visual noise. Visual inspection in this case is passing it through the image decoder.



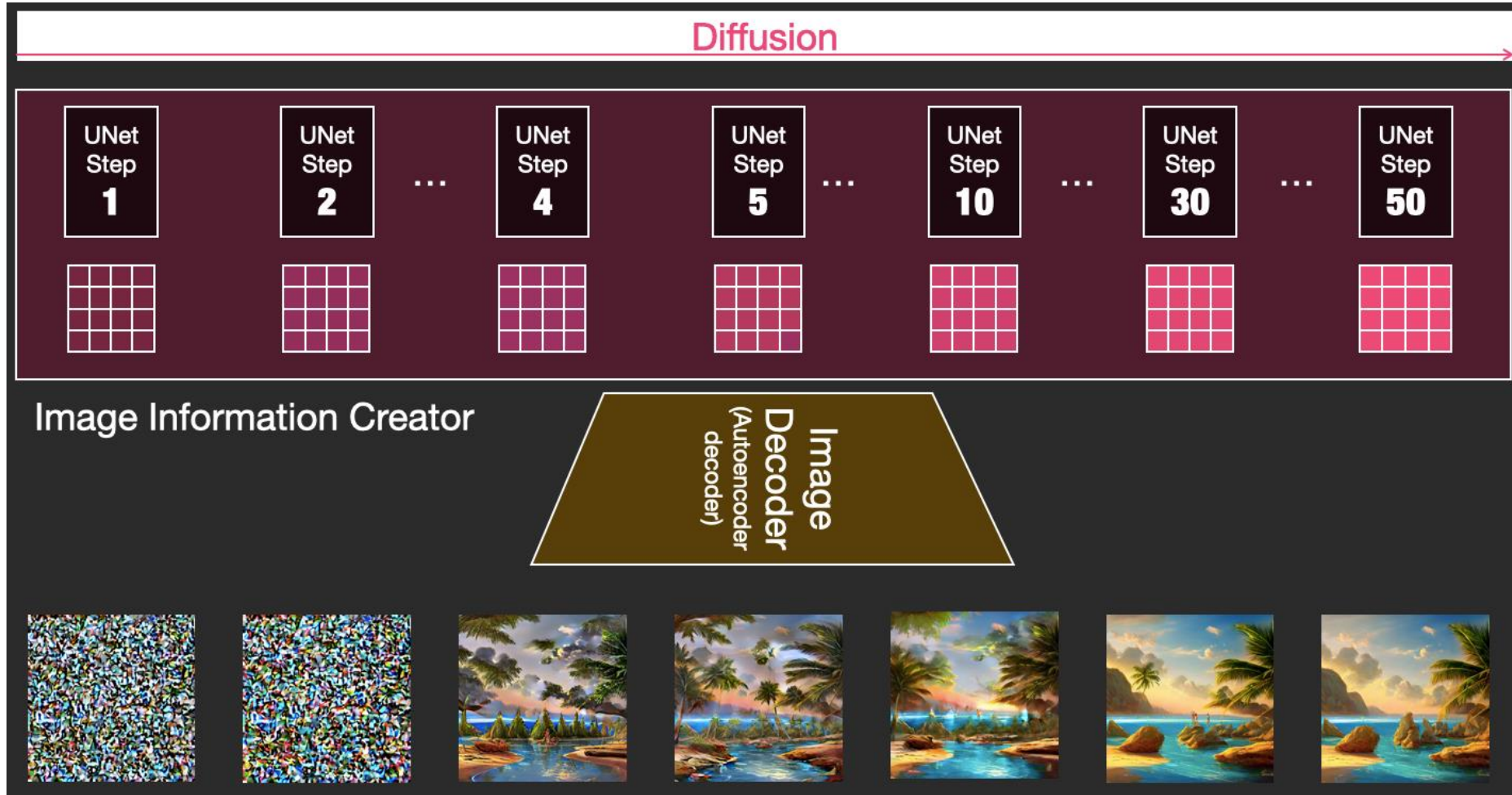
What is Diffusion Anyway?

Diffusion happens in multiple steps, each step operates on an input latents array, and produces another latents array that better resembles the input text and all the visual information the model picked up from all images the model was trained on.



What is Diffusion Anyway?

We can visualize a set of these latents to see what information gets added at each step.



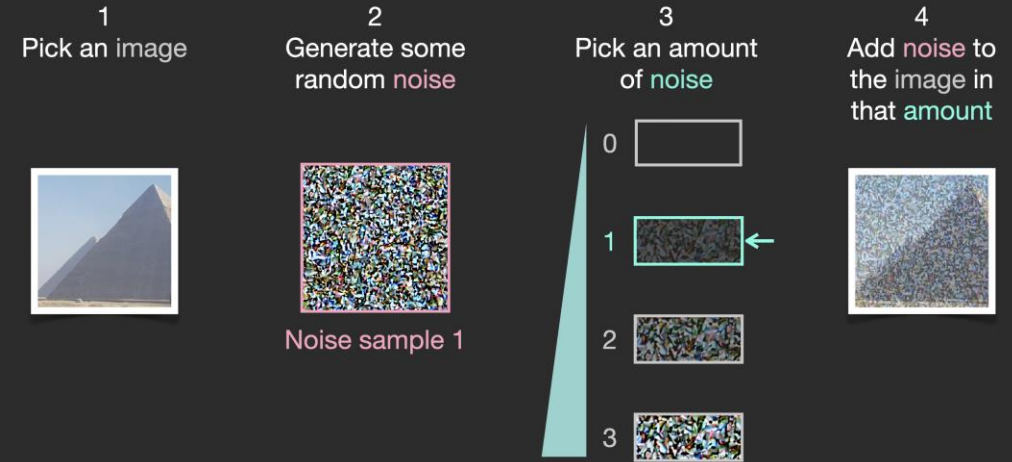
How Diffusion works?

The central idea of generating images with diffusion models relies on the fact that we have powerful computer vision models. Given a large enough dataset, these models can learn complex operations. Diffusion models approach image generation by framing the problem as following:

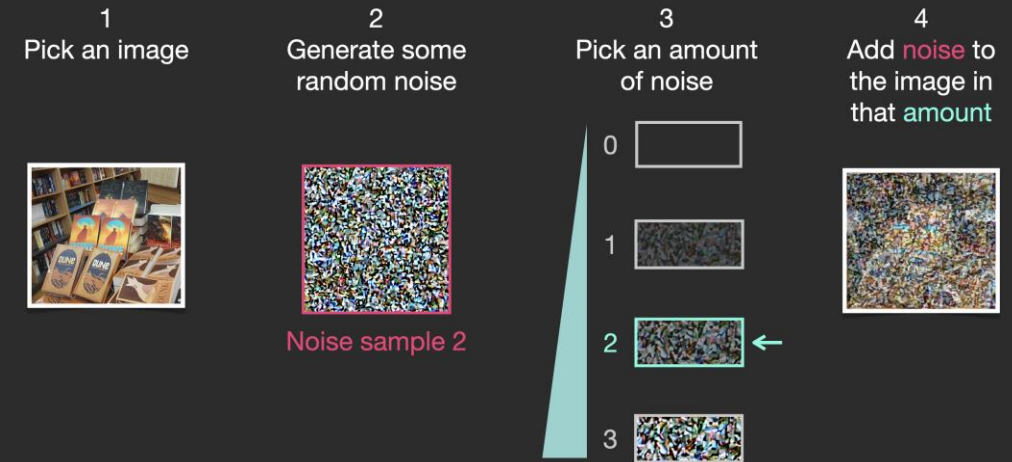
Say we have an image, we generate some noise, and add it to the image

This can now be considered a training example. We can use this same formula to create lots of training examples to train the central component of our image generation model.

Training examples are created by generating **noise** and adding an **amount** of it to the images in the training dataset (forward diffusion)



Generating a 2nd training example with a different image, **noise sample** and **noise amount** (forward diffusion)

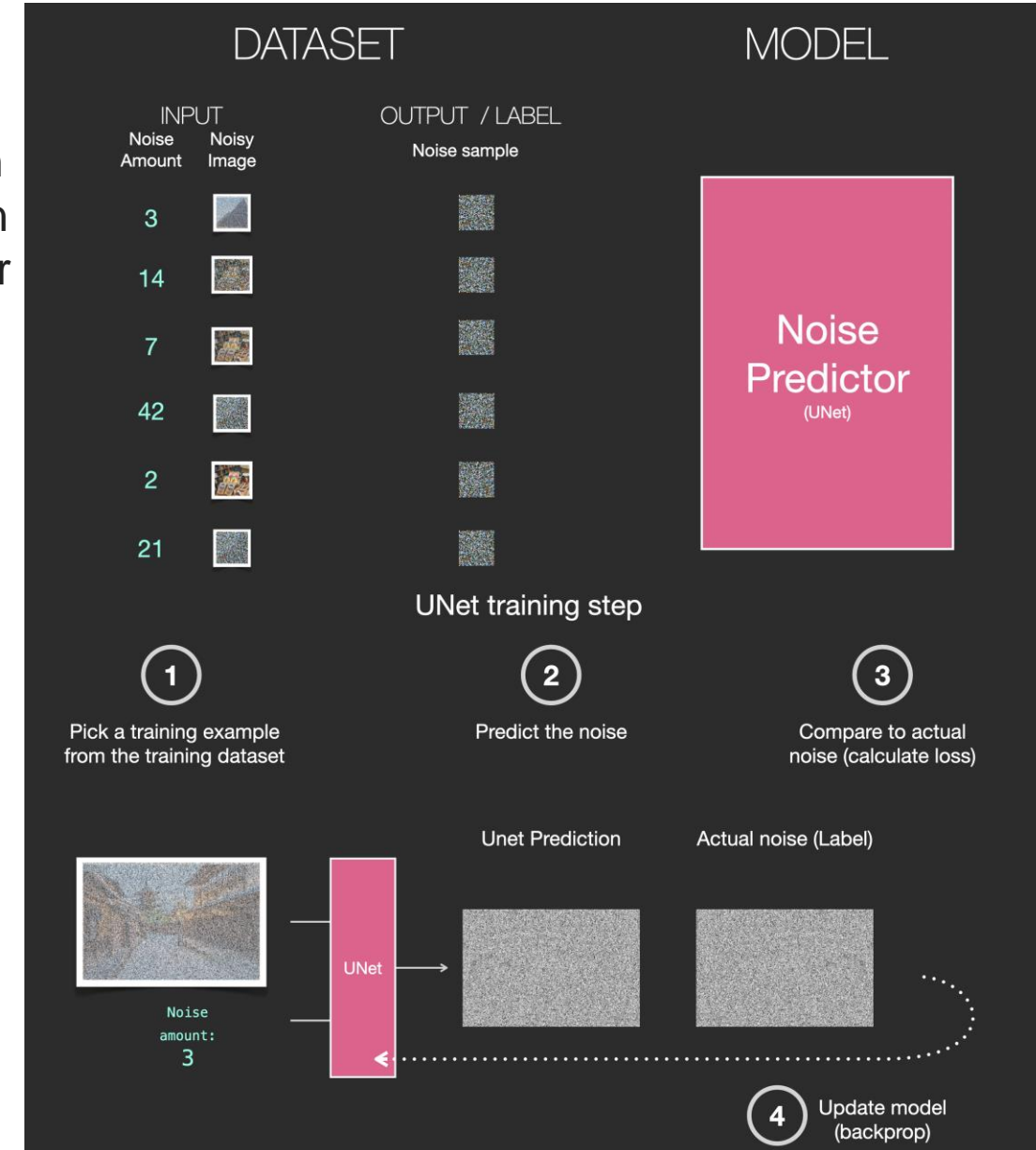


How Diffusion works?

While this example shows a few noise amount values from image (amount 0, no noise) to total noise (amount 4, total noise), we can easily control how much noise to add to the image, and so we can spread it over tens of steps, creating tens of training examples per image for all the images in a training dataset.

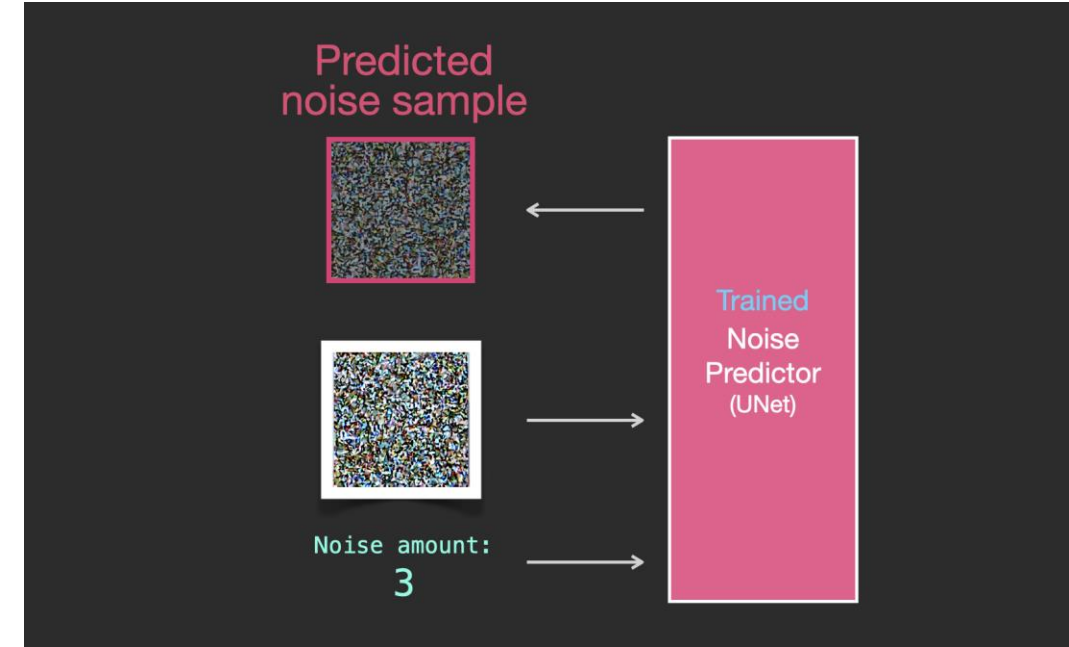
With this dataset, we can train the noise predictor and end up with a great noise predictor that actually creates images when run in a certain configuration.

Now let's see how this generates images

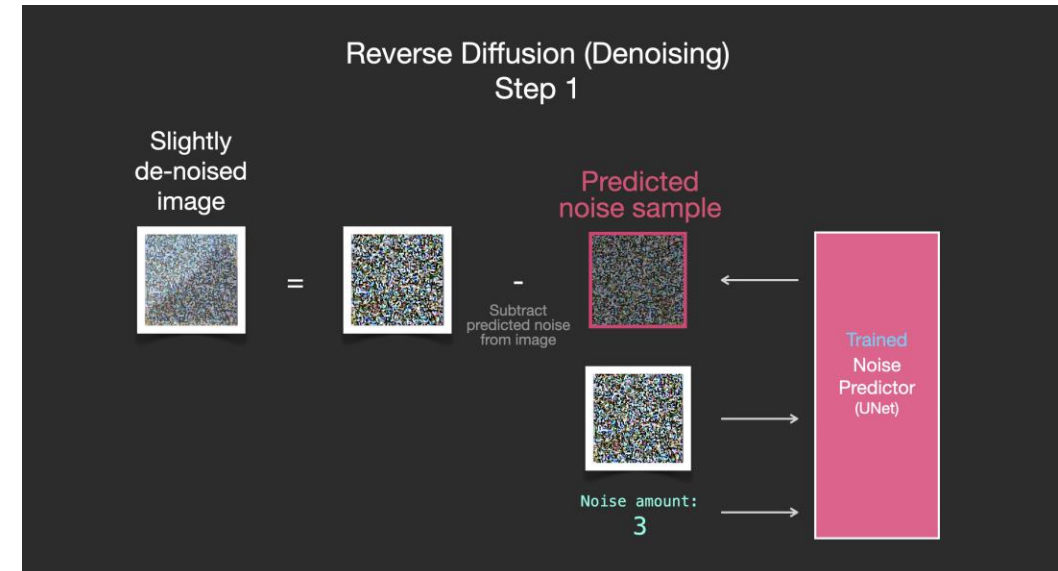


Painting images by removing noise

The trained noise predictor can take a noisy image, and the number of the denoising step, and is able to predict a slice of noise.

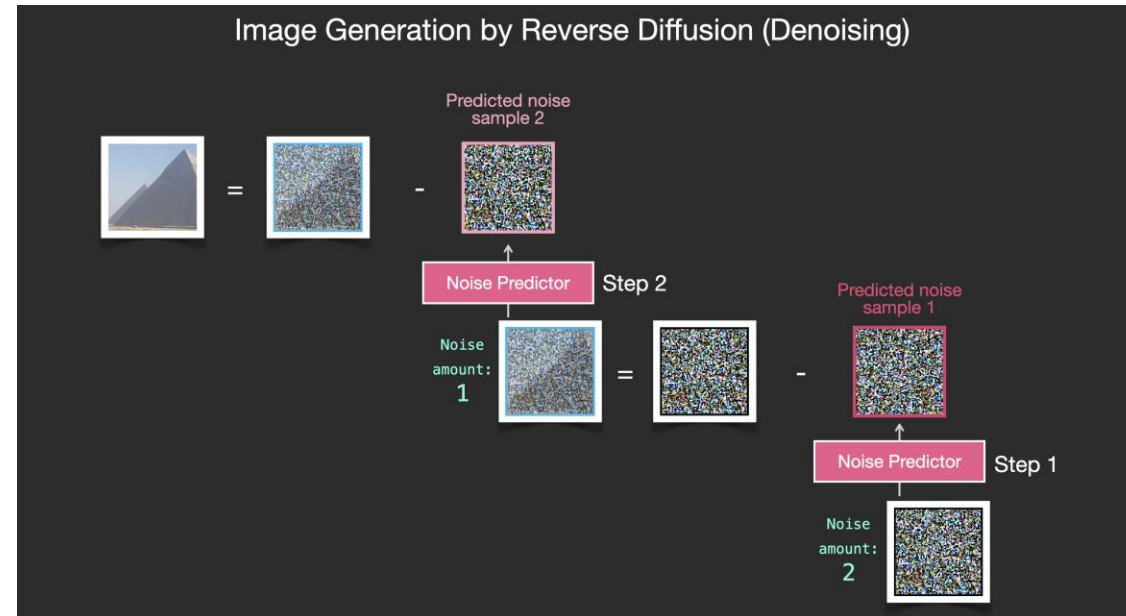


The sampled noise is predicted so that if we subtract it from the image, we get an image that's closer to the images the model was trained on



Painting images by removing noise

If the training dataset was of aesthetically pleasing images (e.g., [LAION Aesthetics](#), which Stable Diffusion was trained on), then the resulting image would tend to be aesthetically pleasing. If we train it on images of logos, we end up with a logo-generating model.



This concludes the description of image generation by diffusion models mostly as described in [Denoising Diffusion Probabilistic Models](#).

Now that you have this intuition of diffusion, you know the main components of not only **Stable Diffusion**, but also **Dall-E 2** and **Google's Imagen**.

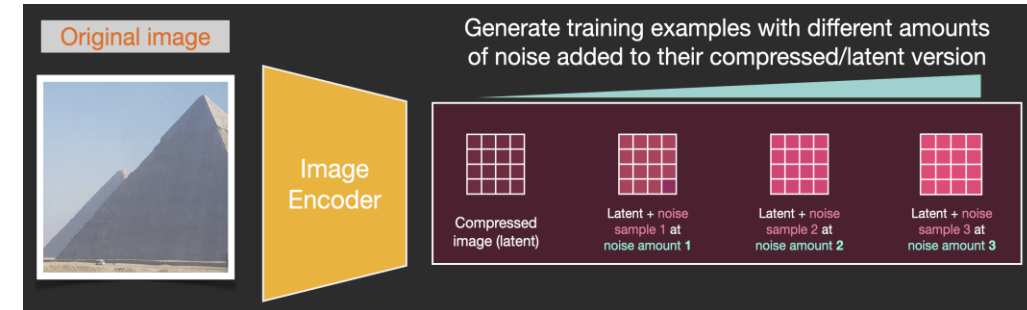
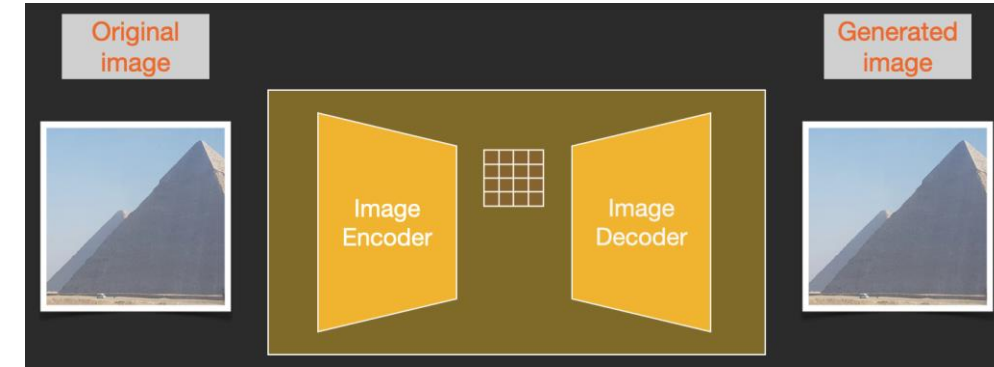
Note that the diffusion process we described so far generates images without using any text data. So if we deploy this model, it would generate great looking images, but we'd have no way of controlling if it's an image of a pyramid or a cat or anything else.

Speed Boost: Diffusion on Compressed (Latent) Data Instead of the Pixel Image

To speed up the image generation process, the Stable Diffusion paper runs the diffusion process not on the pixel images themselves, but on a compressed version of the image. [The paper](#) calls this “Departure to Latent Space”.

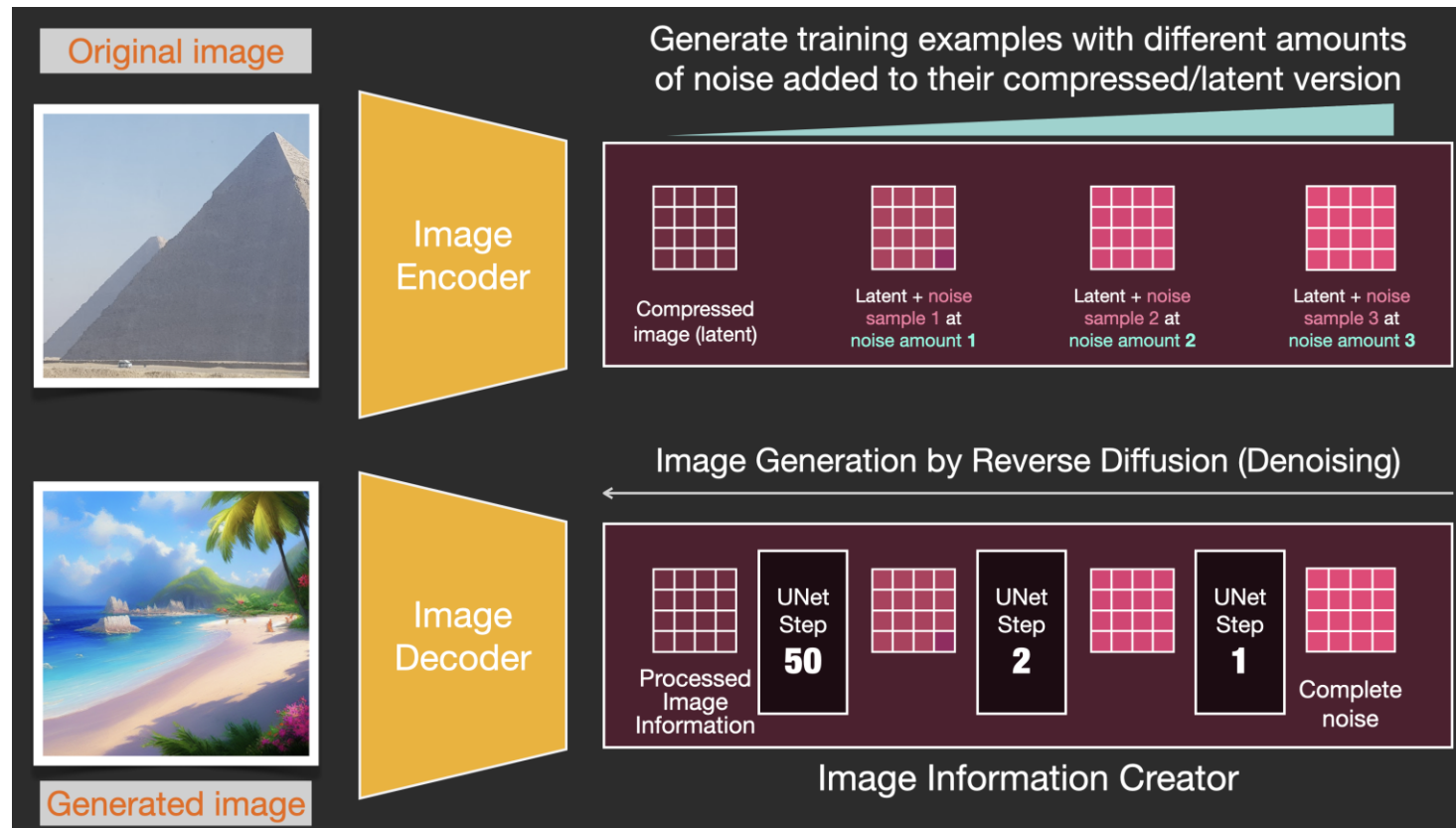
This compression (and later decompression/painting) is done via an autoencoder. The autoencoder compresses the image into the latent space using its encoder, then reconstructs it using only the compressed information using the decoder.

Now the forward diffusion process is done on the compressed latents. The slices of noise are applied to those latents, not to the pixel image. And so the noise predictor is actually trained to predict noise in the compressed representation (the latent space).



Speed Boost: Diffusion on Compressed (Latent) Data Instead of the Pixel Image

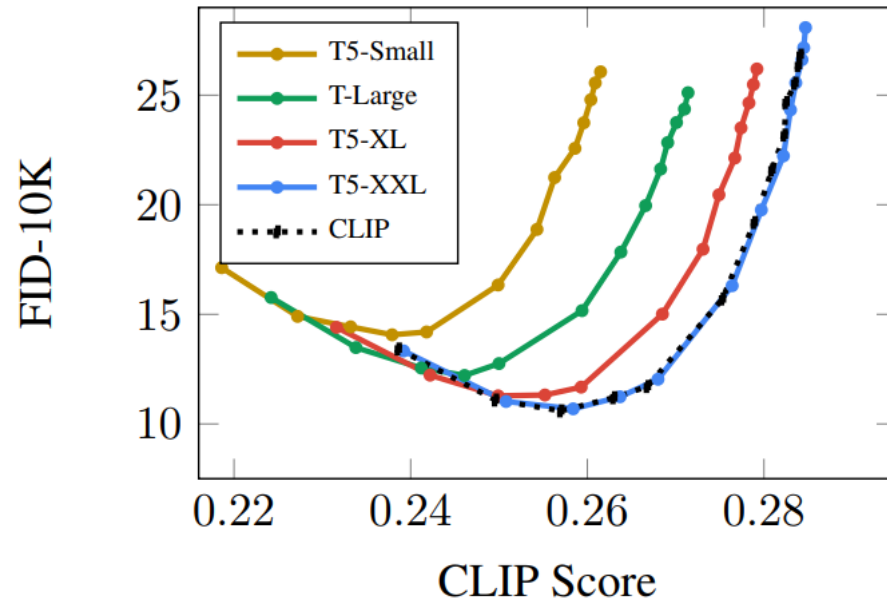
The forward process (using the autoencoder's encoder) is how we generate the data to train the noise predictor. Once it's trained, we can generate images by running the reverse process (using the autoencoder's decoder).



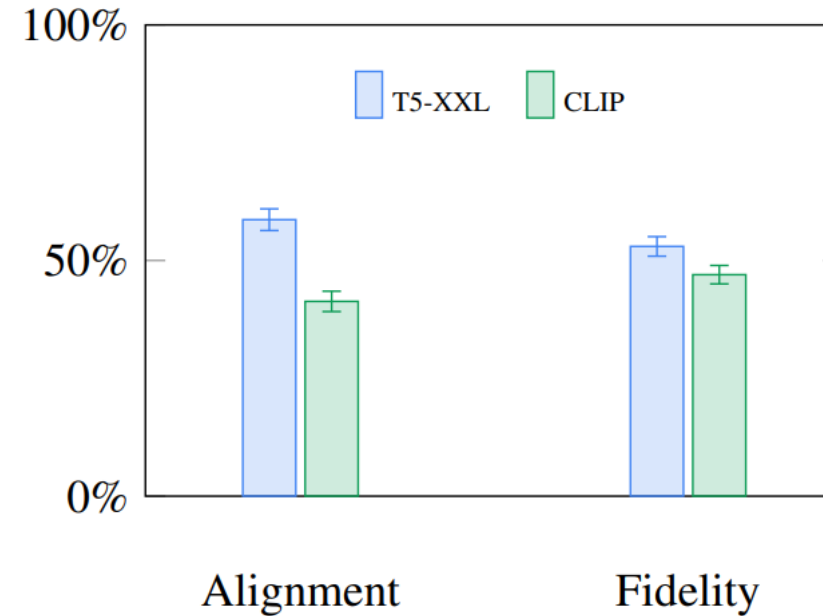
The Text Encoder: A Transformer Language Model

A Transformer language model is used as the language understanding component that takes the text prompt and produces token embeddings. The released Stable Diffusion model uses ClipText then OPENClip (A [GPT-based model](#)), while the paper used [BERT](#).

The choice of language model is shown by the Imagen paper to be an important one. Swapping in larger language models had more of an effect on generated image quality than larger image generation components.






(a) Pareto curves comparing various text encoders.



(b) Comparing T5-XXL and CLIP on DrawBench.

How CLIP is trained

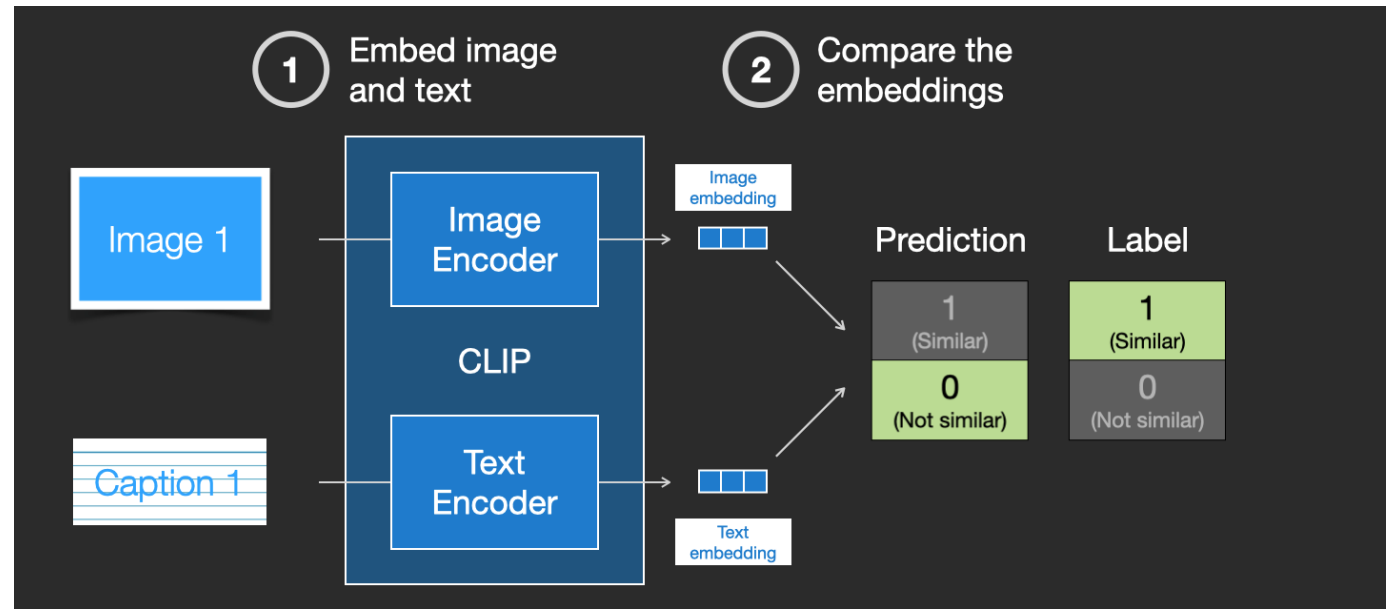
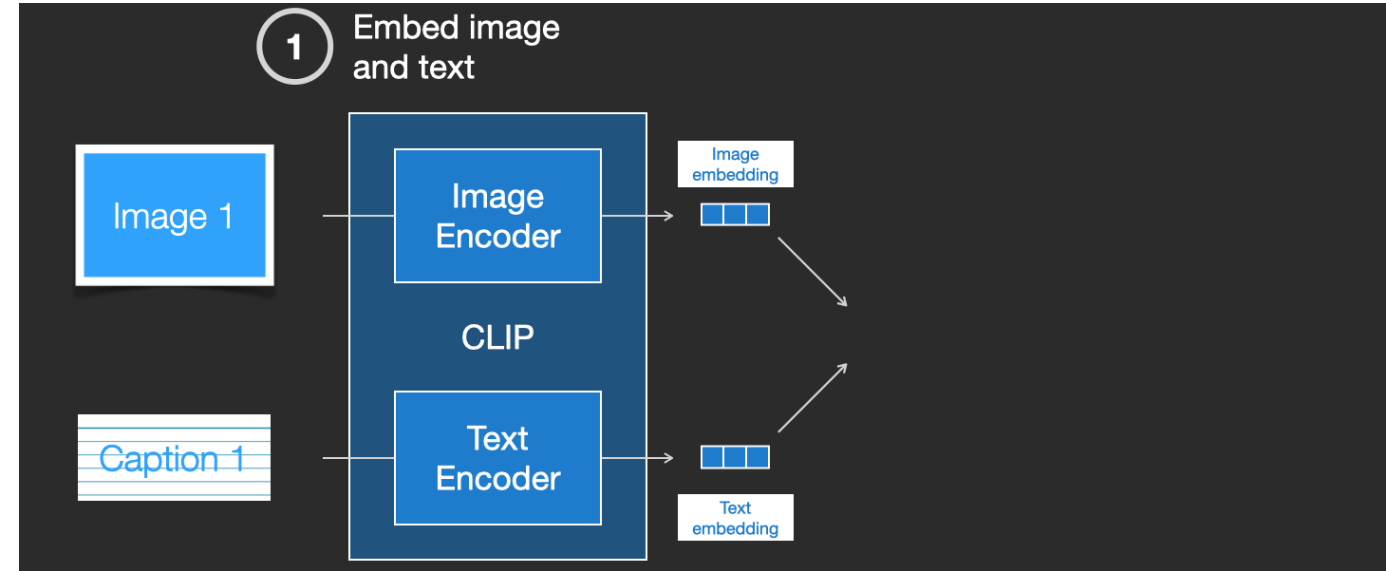
CLIP is trained on a dataset of images and their captions. Think of a dataset looking like this, only with 400 million images and their captions:

IMAGE			
CAPTION	Photo pour Japanese pagoda and old house in Kyoto at twilight - image libre de droit	Soaring by Peter Eades	far cry 4 concept art is the reason why it 39 s a beautiful game vg247. Black Bedroom Furniture Sets. Home Design Ideas

How CLIP is trained

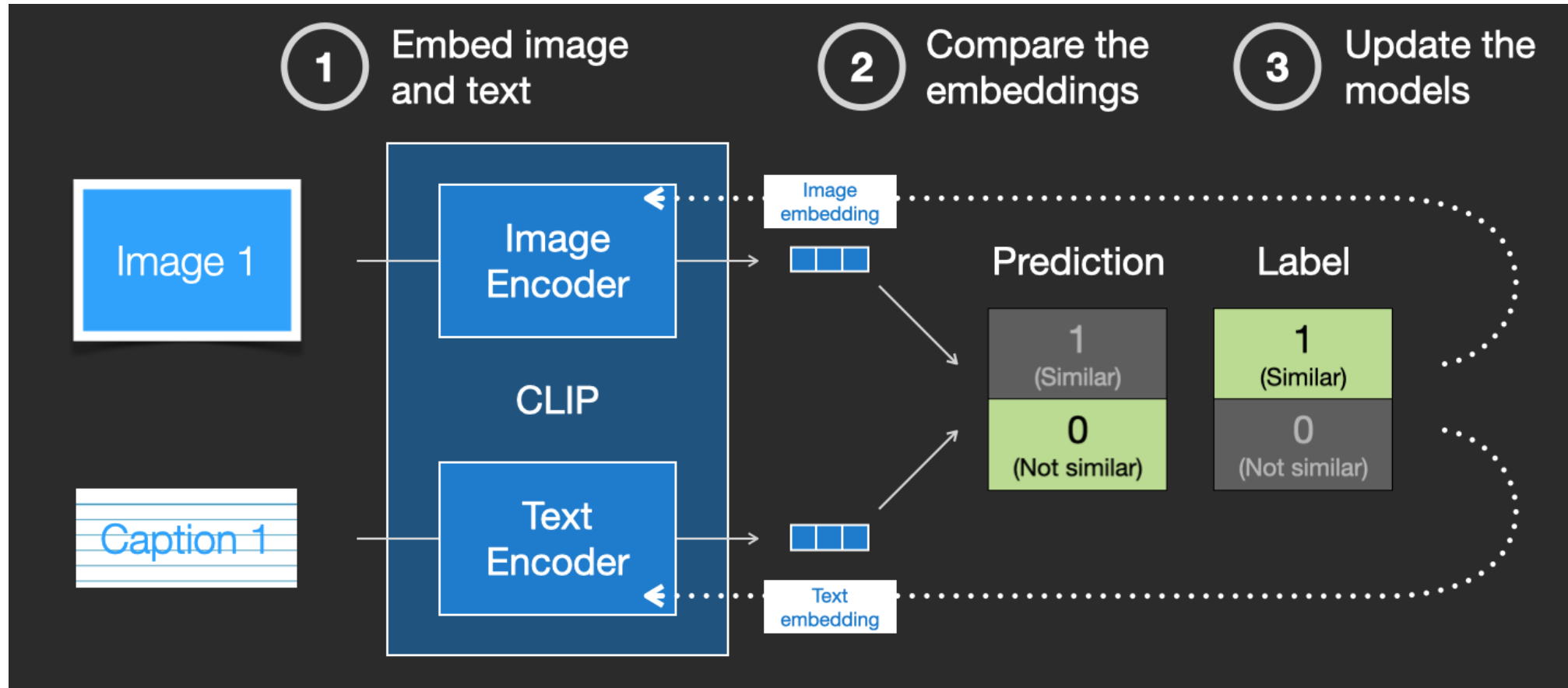
CLIP is a combination of an image encoder and a text encoder. Its training process can be simplified to thinking of taking an image and its caption. We encode them both with the image and text encoders respectively.

We then compare the resulting embeddings using cosine similarity. When we begin the training process, the similarity will be low, even if the text describes the image correctly.



How CLIP is trained

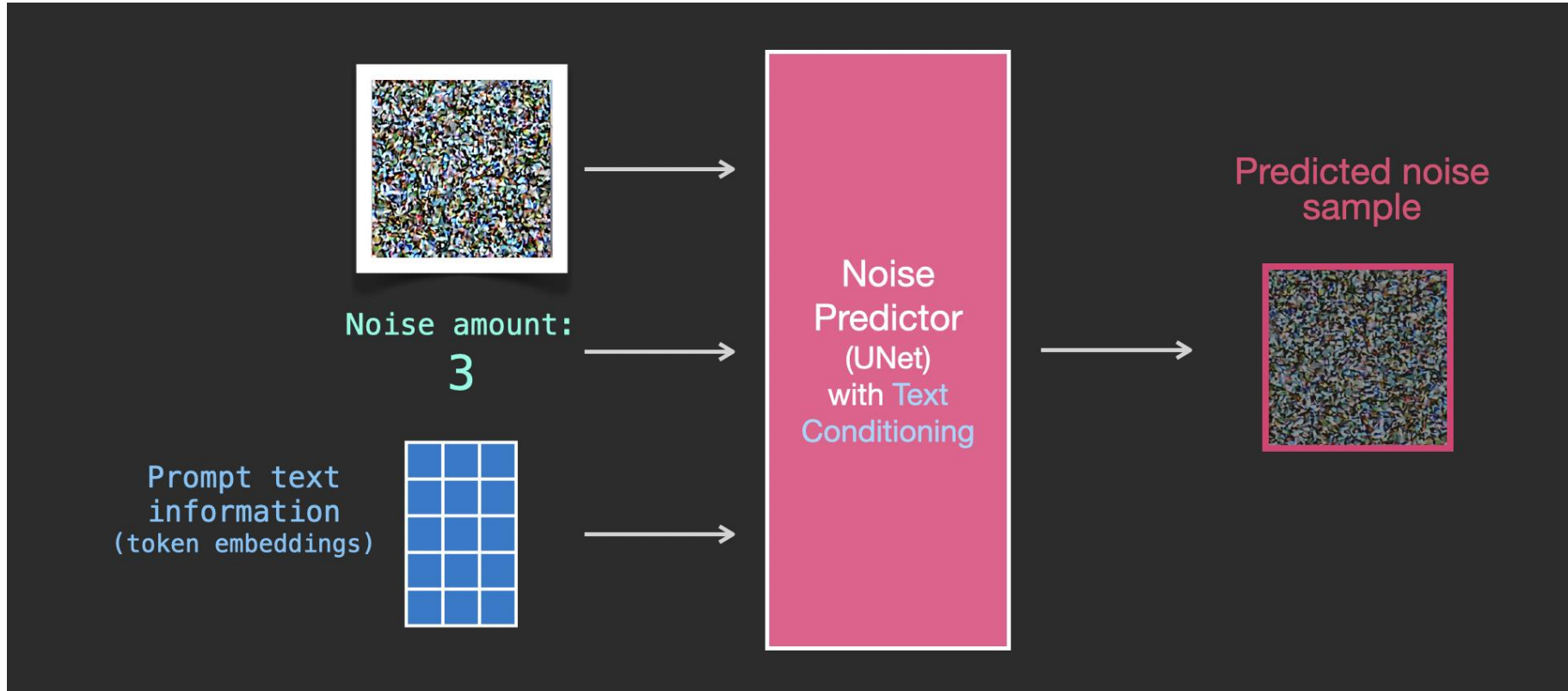
We update the two models so that the next time we embed them, the resulting embeddings are similar.



By repeating this across the dataset and with large batch sizes, we end up with the encoders being able to produce embeddings where an image of a dog and the sentence “a picture of a dog” are similar. Just like in [word2vec](#), the training process also needs to include **negative examples** of images and captions that don’t match, and the model needs to assign them low similarity scores.

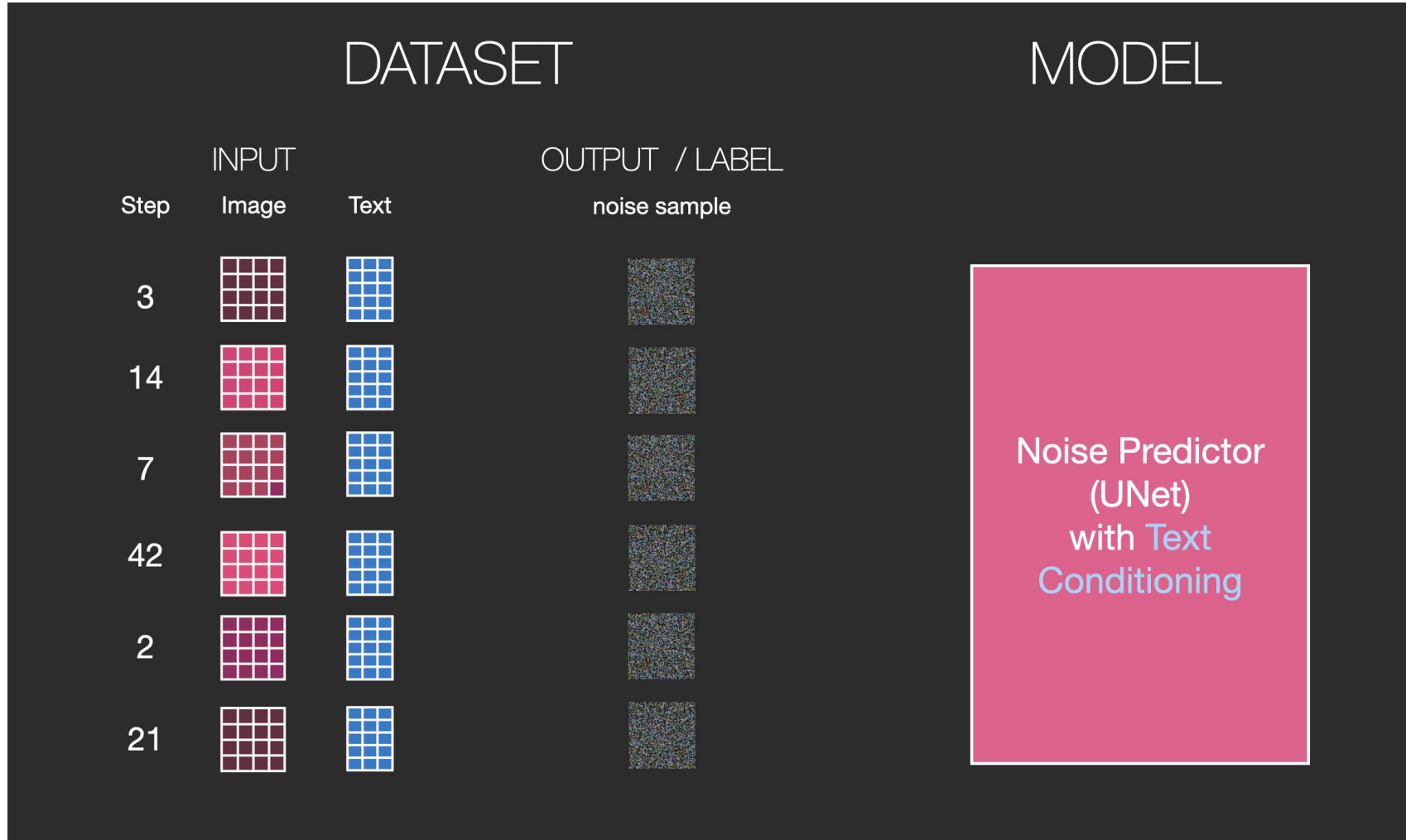
Feeding Text Information Into The Image Generation Process

To make text a part of the image generation process, we have to adjust our noise predictor to use the text as an input.



Feeding Text Information Into The Image Generation Process

Our dataset now includes the encoded text. Since we're operating in the latent space, both the input images and predicted noise are in the latent space.



Newer models from stability.ai: SDXL

Feature	Stable Diffusion 1 (SD1)	Stable Diffusion XL (SDXL)
Model Size	~860M U-Net, 4.1B CLIP	3.5B U-Net, 6.6B total
Architecture	Single U-Net	Dual U-Net (Base + Refiner)
Default Resolution	512x512	1024x1024
Image Quality	Decent, but struggles with details like hands	Much more realistic, detailed, and high-quality
Text Encoder	OpenAI's CLIP	OpenCLIP ViT-G/14 (better prompt understanding)
Prompt Understanding	Struggles with complex prompts	More accurate and nuanced interpretation
Performance	Runs on lower-end GPUs (8GB VRAM)	Requires more VRAM (12GB+, ideally 16GB)
Customization	More community-trained LoRAs & models	Newer, but supports LoRA and fine-tuning
Inpainting & Editing	Requires extra models	Built-in improvements for inpainting
Best For	Casual use, lightweight AI art	High-quality, professional-grade AI images