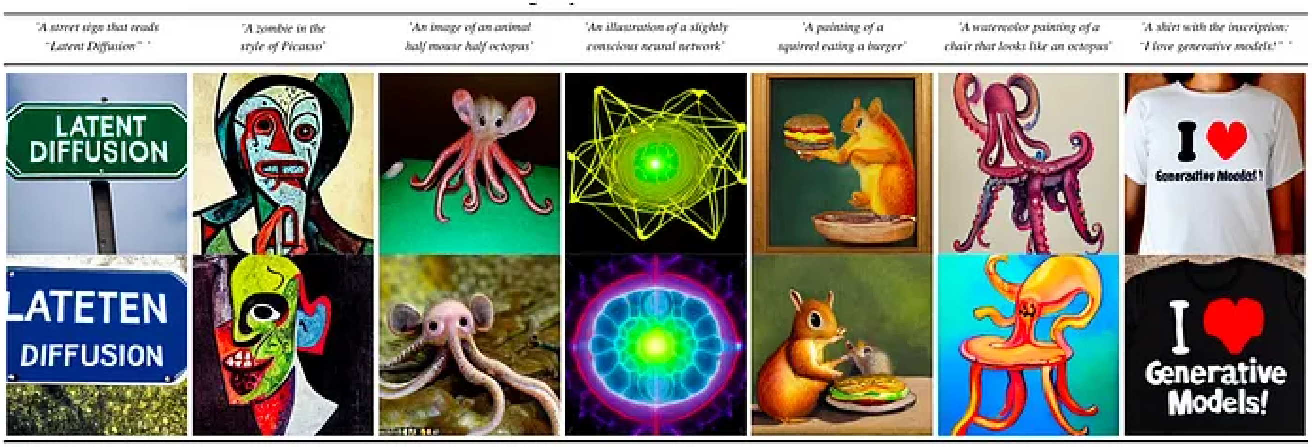
Stable Diffusion Explained

How does Stable diffusion work? Explaining the tech behind text to image generation.

Onkar Mishra · Follow
6 min read · Jun 8, 2023



User defined text prompts for text to image synthesis

Large text to image models have achieved remarkable success in enabling high quality synthesis of images from text prompts. Diffusion models can be applied to text to image generation tasks to achieve state of art image generating results.

Stable Diffusion model has achieved state of the art results for image generation. Stable Diffusion is based on a particular type of diffusion model called **Latent Diffusion model**, proposed in High-Resolution Image Synthesis with Latent Diffusion Models and created by the researchers and engineers from CompVis, LMU and RunwayML. The model was initially trained on 512x512 images from a subset of the LAION-5B database.

This is particularly achieved by encoding text inputs into latent vectors using pretrained language models like CLIP. Diffusion models can achieve state-of-the-art results for generating image data from texts. But the process of denoising is very slow and consumes a lot of memory when generating high-resolution images. Therefore, it is challenging to train these models and also use them for inference.

In this regard, latent diffusion can reduce the memory and computational time by applying the diffusion process over a lower dimensional *latent* space, instead of using the actual pixel space. In latent diffusion, the model is trained to generate latent (compressed) representations of the images.

Training of Diffusion Model

Stable Diffusion is a large text to image diffusion model trained on billions of images. Image diffusion model learn to denoise images to generate output images. Stable Diffusion uses latent images encoded from training data as input. Further, given an image zo, the diffusion algorithm progressively add noise to the image and produces a noisy image zt, with t being how many times noise is added. When t is large enough, the image approximates pure noise. Given a set of inputs such as time step t, text prompt, image diffusion algorithms learn a network to predict the noise added to the noisy image zt.

There are mainly 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. During latent diffusion *training,* the encoder converts a 512*512*3 image into a low dimensional latent representation of image of size say 64*64*4 for the forward diffusion process. We call these small encoded versions of images as **latents**. We apply more and more noise to these latents at each step of training. This encoded latent representation of images acts as the input to the *U-Net* model.

Here, we are converting an image of shape `(3, 512, 512)` into a latent of shape `(4, 64, 64)`, which requires 48 times less memory. This leads to reduced memory and compute requirements compared to pixel-space diffusion models. Thus, we are able to generate `512 × 512` images very quickly on 16GB Colab GPUs as well.

Top highlight

The decoder transforms the latent representation back into an image. We convert the denoised latents generated by the reverse diffusion process into images using the VAE decoder.

During *inference,* we only need the VAE decoder to convert the denoised image into actual images.

```python
from torchvision import transforms as tfms
from diffusers import AutoencoderKL

# Load the autoencoder model which will be used to decode the latents into image
vae = AutoencoderKL.from_pretrained("CompVis/stable-diffusion-v1-4", subfolder="

# To the GPU we go!
vae = vae.to(torch_device)

# Convert PIL image to latents

def pil_to_latent(input_im):
    # Single image -> single latent in a batch (so size 1, 4, 64, 64)
    with torch.no_grad():
        latent = vae.encode(tfms.ToTensor()(input_im).unsqueeze(0).to(torch_devi
    return 0.18215 * latent.latent_dist.sample()
```

2. UNet

The U-Net predicts denoised image representation of noisy latents. Here, noisy latents act as input to Unet and the output of UNet is noise in the latents. Using this, we are able to get actual latents by subtracting the noise from the noisy latents.

The Unet that takes in the noisy latents (x) and predicts the noise. We use a conditional model that also takes in the timestep (t) and our text embedding as guidance.

Thus, the model looks like this:

```
from diffusers import UNet2DConditionModel

# The UNet model for generating the latents.
unet = UNet2DConditionModel.from_pretrained("CompVis/stable-diffusion-v1-4", sub

# To the GPU
unet = unet.to(torch_device);
noise_pred = unet(latents, t, encoder_hidden_states=text_embeddings)["sample"]
```

The model is essentially a UNet with an encoder(12 blocks), a middle block and a skip connected decoder(12 blocks). In these 25 blocks, 8 blocks are down sampling or upsampling convolution layer and 17 blocks are main blocks that each contain four resnet layers and two Vision Transformers(ViTs). Here 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.

3. The Text-encoder

The text-encoder transforms the input prompt into an embedding space that goes as input to the U-Net. This acts as guidance for noisy latents when we train Unet for its denoising process. The text encoder is usually a simple *transformer-based* encoder that maps a sequence of input tokens to a sequence of latent text-embeddings. Stable Diffusion does not train a new text encoder and instead uses an already trained text encoder, CLIP. The text encoder creates embeddings corresponding to the input text.

Tokenization

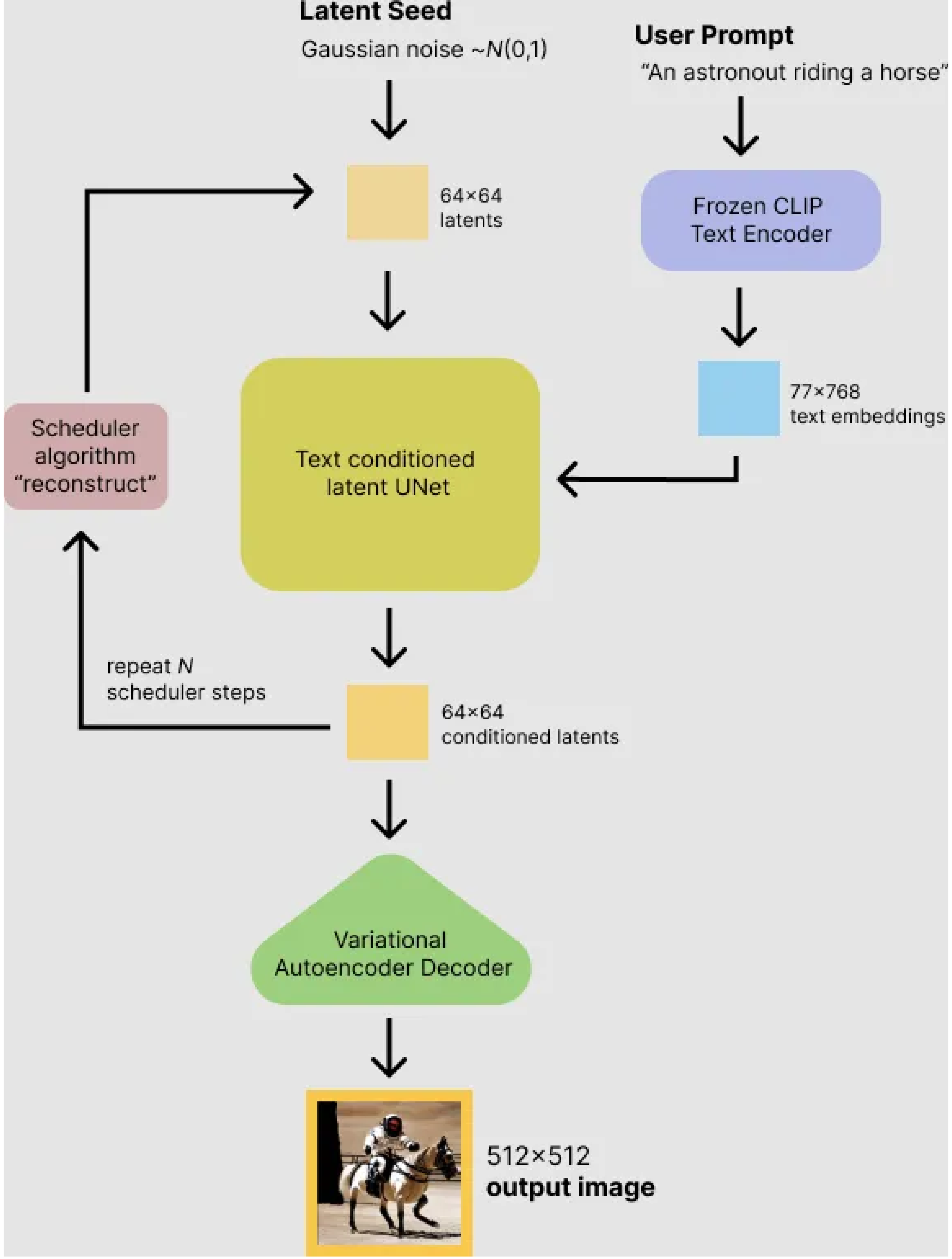
```
from transformers import CLIPTextModel, CLIPTokenizer

# Load the tokenizer and text encoder to tokenize and encode the text.
tokenizer = CLIPTokenizer.from_pretrained("openai/clip-vit-large-patch14")
text_encoder = CLIPTextModel.from_pretrained("openai/clip-vit-large-patch14")

# To the GPU
text_encoder = text_encoder.to(torch_device)

prompt = 'An astronaut riding a horse'
# Turn the text into a sequnce of tokens:
text_input = tokenizer(prompt, padding="max_length", max_length=tokenizer.model_
input_ids = text_input.input_ids.to(torch_device)
```

Output Embedding

```
# Get output embeddings from tokens
output_embeddings = text_encoder(text_input.input_ids.to(torch_device))[0]
```

```
print('Shape:', output_embeddings.shape)
```

Putting it all together, the model works as follow during inference process:



Latent Seed
Gaussian noise ~N(0,1)
User Prompt
"An astronout riding a horse"
64×64
latents
Frozen CLIP
Text Encoder
77×768
text embeddings
Scheduler
algorithm
"reconstruct"
Text conditioned
latent UNet
repeat N
scheduler steps
64×64
conditioned latents
Variational
Autoencoder Decoder
512×512
output image

Scheduler

Apart from above 3, we have Scheduler which is used to add noise to an image and then use model to predict the noise.

```
from diffusers import LMSDiscreteScheduler
scheduler = LMSDiscreteScheduler(beta_start=0.00085, beta_end=0.012, beta_schedu
```

Above sets up a scheduler used to train the model. In case, we want to set up a scheduler for smaller number of steps, we set up scheduler as follow:

```
# Set the number of sampling steps:
scheduler.set_timesteps(15)
```

Latent Diffusion Model like Stable Diffusion enable various creative applications like:

1. Text-to-Image Generation

2. Image-to-Image Generation — Generate or modify new images based on a starting point
3. Image Upscaling — Enlarge an image into larger image
4. Inpainting — Modify a specific area of an image by masking out the area and then generating new details on the area based on a provided prompt.

Latent Diffusion Model also reduces the cost of training and inference that have the potential to democratise high resolution image synthesis to masses.

In my next blog, I will be discussing about textual inversion, which is a technique to fine tune Stable Diffusion to learn a novel concept or task.

Reference:

1. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10684–10695).
2. Zhang, L., & Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*.
3. https://huggingface.co/docs/diffusers/index

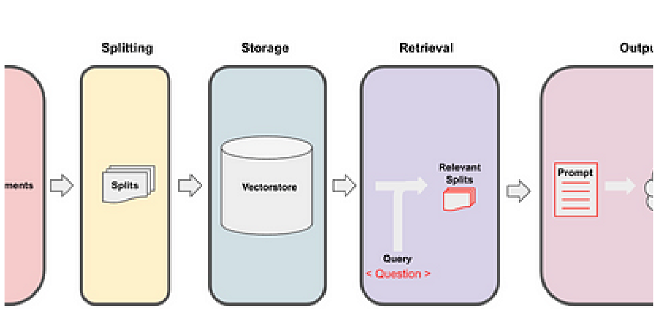
Stable Diffusion | Ai Art | Image Processing | AI | Generative Art

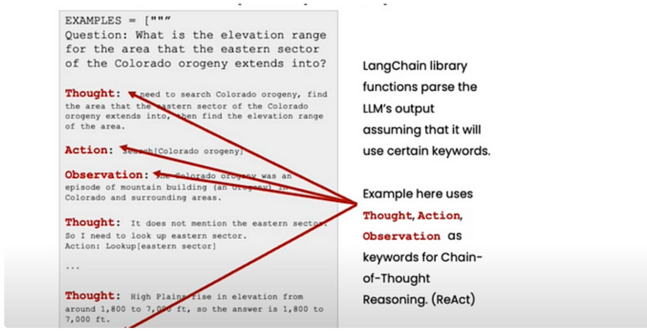
Written by Onkar Mishra

822 Followers

Follow

More from Onkar Mishra





Onkar Mishra

Using langchain for Question Answering on own data

Step-by-step guide to using langchain to chat with own data

23 min read · Aug 7, 2023

1.2K 14

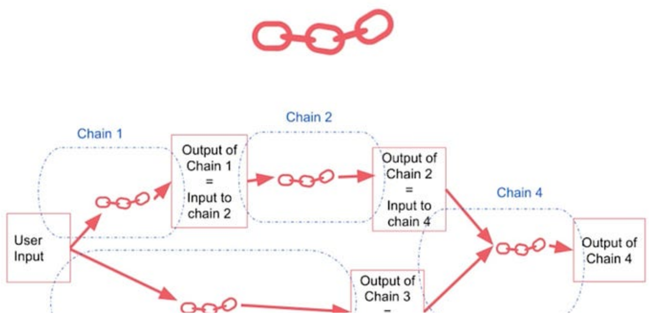
Onkar Mishra

Using langchain for large language model application development

Step by step guide to create LLM applications using langchain

8 min read · Jul 24, 2023

277 1



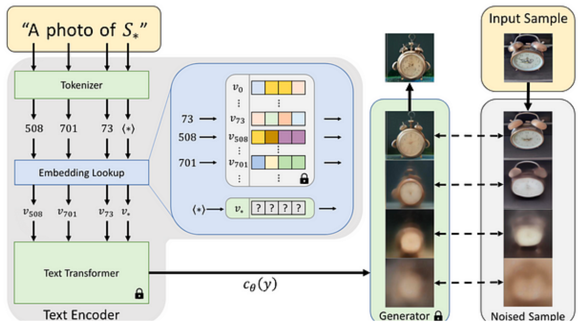
Onkar Mish... in Artificial Intelligence in Plain Engl...

Using Chains and Agents for LLM application development

Step by step guide to use chains and agents in langchain

16 min read · Aug 2, 2023

449 1



Onkar Mishra

Textual Inversion: A method to finetune Stable Diffusion Model

How textual inversion works and application of textual inversion in image synthesis

9 min read · Jun 13, 2023

357 1

See all from Onkar Mishra

Recommended from Medium

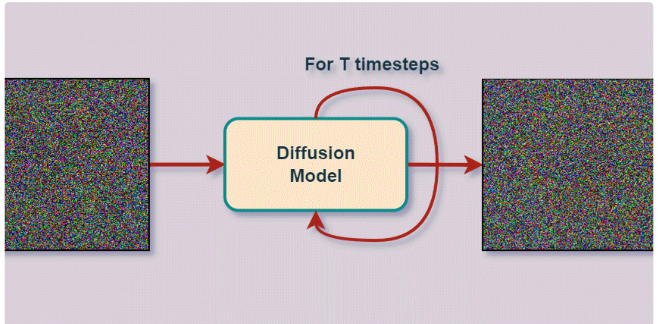
Jonathan Kernes in Towards Data Science

Diffusion Models

What are they, how do they work, and why now?

23 min read · Dec 13, 2022

290 4



Gabriel Mongaras in Better Programming

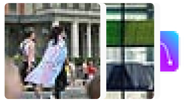
Diffusion Models—DDPMs, DDIMs, and Classifier Free Guidance

A guide to the evolution of diffusion models from DDPMs to Classifier Free guidance

28 min read · Mar 13, 2023

616 8

Lists



Generative AI Recommended Reading

52 stories · 639 saves

What is ChatGPT?

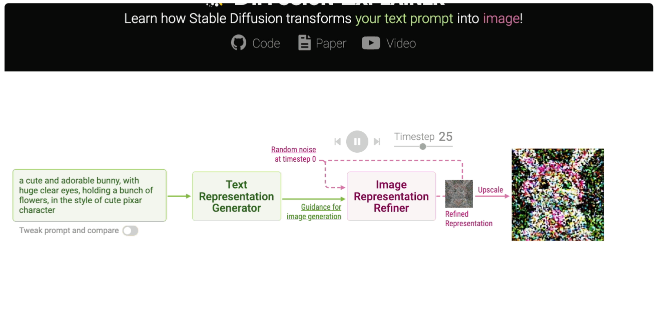
9 stories · 278 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 276 saves

Tech & Tools

16 stories · 133 saves



Seongmin ... in Polo Club of Data Science | Georgi...

Stable Diffusion Explained and Visualized for Everyone

4 min read · Nov 11, 2023

172

Naman Rastogi

Navigating DDPMs—A Closer Look At Denoising Diffusion...

In the realm of nature's uncertainty, probability becomes the beacon that...

28 min read · Aug 17, 2023

114 1

Aguimar Neto

What is Latent Diffusion in AI?

Latent diffusion models are deep learning models that have recently emerged as a...

5 min read · Oct 7, 2023

2

Sarit Ritwirune

Stable Diffusion openpose with wood figure experiment

The original coversation comes from Stable Diffusion Thailand

2 min read · Aug 3, 2023

1

See more recommendations