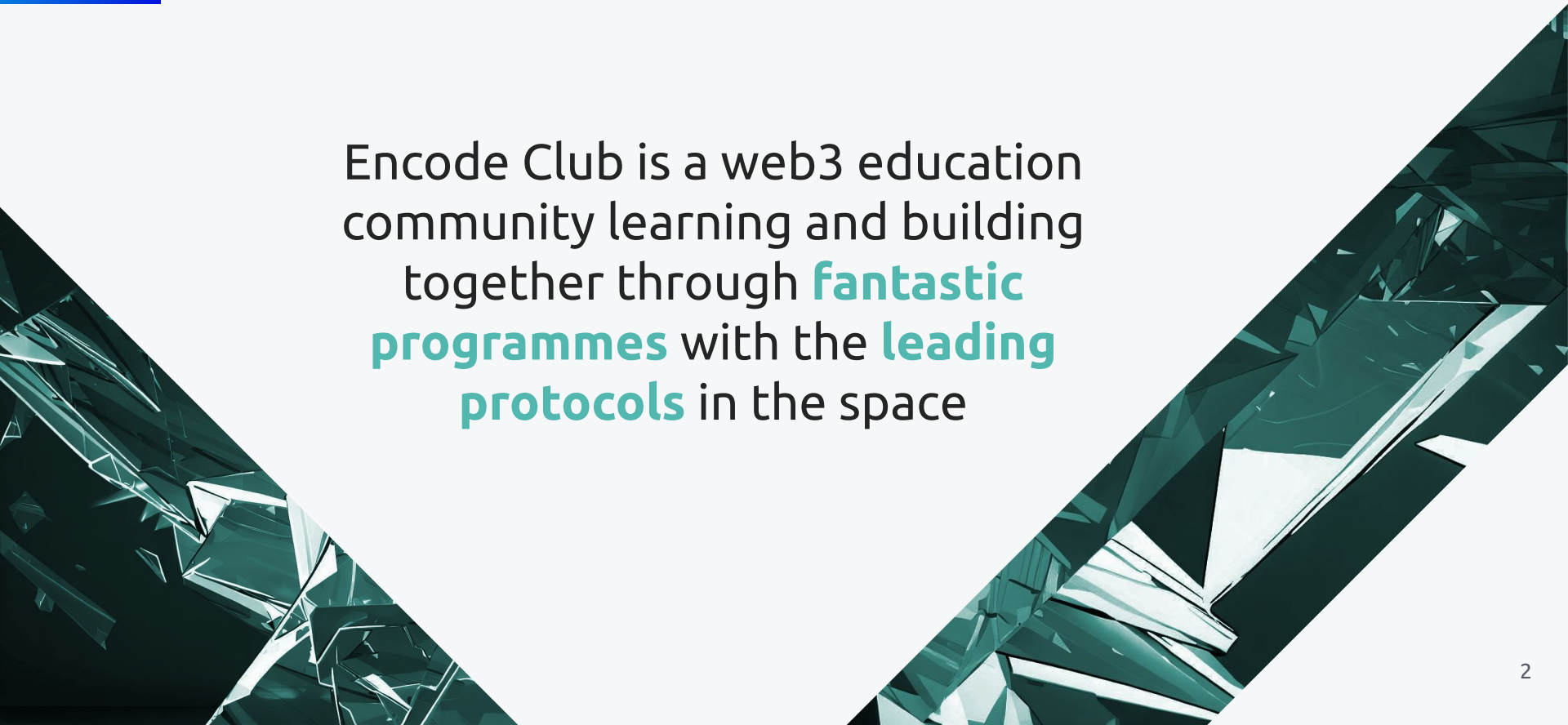
VENTURE MINER

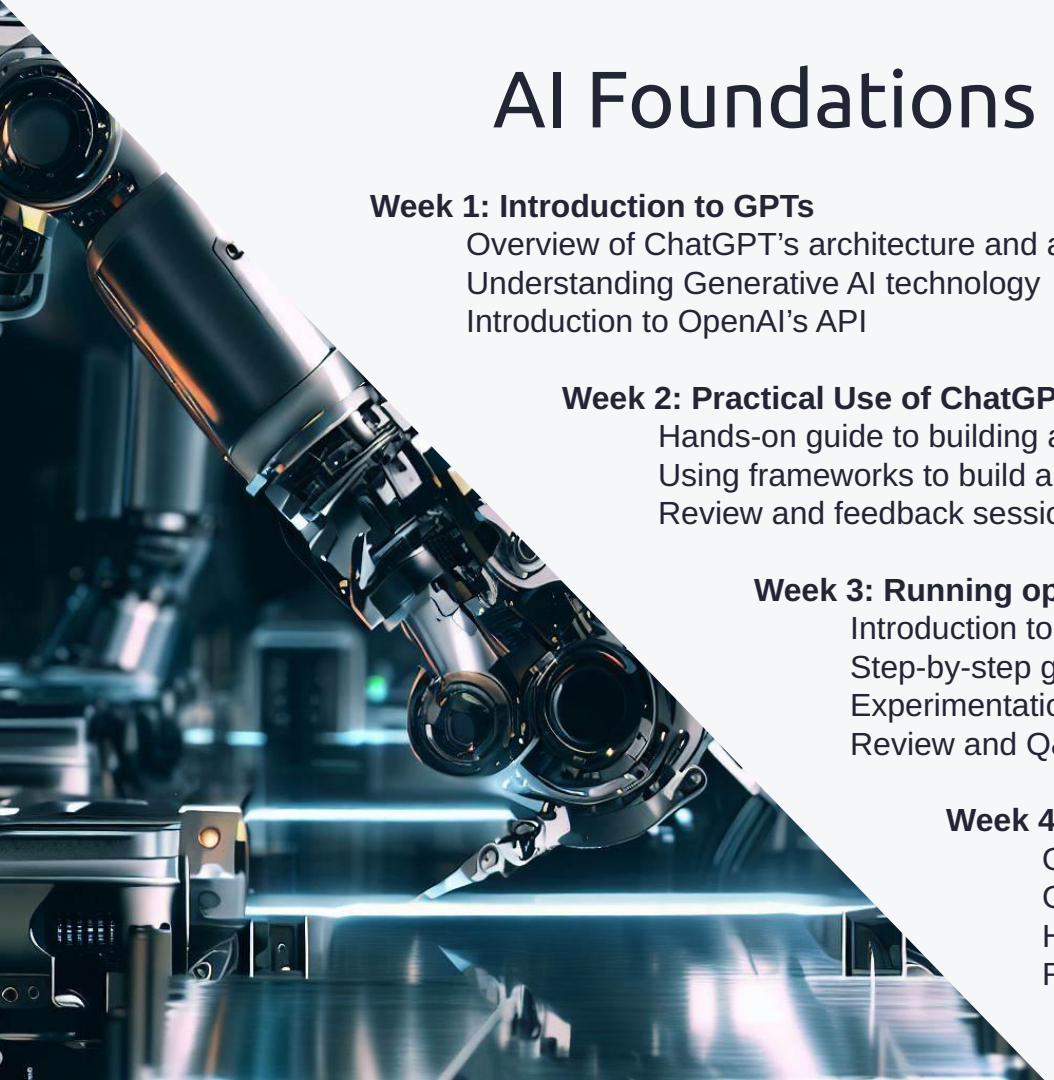
Early stage Web3 and AI
Venture Studio

In partnership with **Encode Club**

Encode Club is a web3 education community learning and building together through **fantastic programmes** with the **leading protocols** in the space



AI Foundations Bootcamp



Week 1: Introduction to GPTs

- Overview of ChatGPT’s architecture and applications
- Understanding Generative AI technology
- Introduction to OpenAI’s API

Week 2: Practical Use of ChatGPT

- Hands-on guide to building a simple Chat app with OpenAI’s API
- Using frameworks to build a simple Chat app
- Review and feedback session on ChatGPT capabilities

Week 3: Running open models locally

- Introduction to open LLMs
- Step-by-step guide to run local LLMs
- Experimentation with text-to-text tasks
- Review and Q&A on running local models

Week 4: Generating Images with Python

- Overview of image generation techniques
- Guide to executing scripts for image generation
- Hands-on experimentation with image generation
- Review and Q&A on image generation techniques

VENTURE MINER

Generating Images

Topics

- Generative AI review
- Image generation overview
    - Introduction
    - Diffusion
    - Stable diffusion
    - Sampling
    - Models comparison
- Hands-on experimentation
    - Setting up the environment
    - Running a model with Python
- Running models with a local WebUI
    - Cloning and installation
    - Using the Stable Diffusion WebUI
- Experimentation
    - Models
    - Checkpoints
    - Sampling methods
    - Configurations
    - Prompts
- Conclusion

VENTURE MINER

Review

Generative AI

- How models work
  - Data
  - Training
  - Fine tuning
  - Prompting
- Using APIs or running locally
  - Hardware requirements
  - Maintenance
  - Implementation
  - Flexibility
- Using a WebUI to run models
  - Installation
  - Executing
  - Prompting models
  - API calls

Review

Environment setup for Stable Diffusion tools

- Install all the Dependencies for *AUTOMATIC1111* Stable Diffusion WebUI
https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki/Dependencies
- Install *AUTOMATIC1111* Stable Diffusion WebUI with the correct dependencies for your GPU
    - NVidia:
https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki/Install-and-Run-on-NVidia-GPUs
    - AMD:
https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki/Install-and-Run-on-AMD-GPUs
    - Intel:
https://github.com/openvinotoolkit/stable-diffusion-webui/wiki/Installation-on-Intel-Silicon
- Make sure that you can run and access the app in your localhost

Image generation overview

Introduction

Modern AI models for image generation works somewhat similar to what we've learned of GPT LLMs.

They often utilize a similar **transformer-based** generative architecture **adapted** for visual data.

Instead of processing and generating textual sequences, these models handle **pixel** or **feature** sequences.

The process begins with a **text prompt** or partially completed **image**, which is encoded into a *latent space representation* akin to how GPTs encode textual input.

The transformer then **iteratively refines** this representation, guided by learned patterns from its training.

VENTURE MINER

Image generation overview

Introduction

The same way that a GPT can **infer tokens** related to other tokens, an image generation model can **infer pixels** and **features** related to other pixels and features.

Starting from a completely **unintelligible** image, in each step it predicts and adds details, eventually decoding the latent representation into a more **coherent detailed** image.

This approach allows the model to generate highly detailed and contextually relevant images from **textual descriptions** or **partial images**, leveraging the transformer's ability to handle complex dependencies and relationships, just like GPT models do with text.

VENTURE MINER

Image generation overview

Diffusion

Imagine you're wearing your favorite shirt and accidentally spill coffee on it. At first, the coffee stain is just a confined, **concentrated spot**, randomly placed over the fabric.

As time passes, the stain starts to **spread**, becoming fainter but covering more of the shirt. This spreading is like **diffusion** - where particles (or in this case, coffee molecules) move from an area of higher concentration to lower concentration, eventually spreading out evenly.

In a similar manner, diffusion models starts with clear data (unstained shirt) and then introduces **noise** (coffee spill).

As the model progresses, this noise **spreads** and **diffuses** through the data, much like the coffee stain is slowly dispersed across the fabric, altering its features completely.

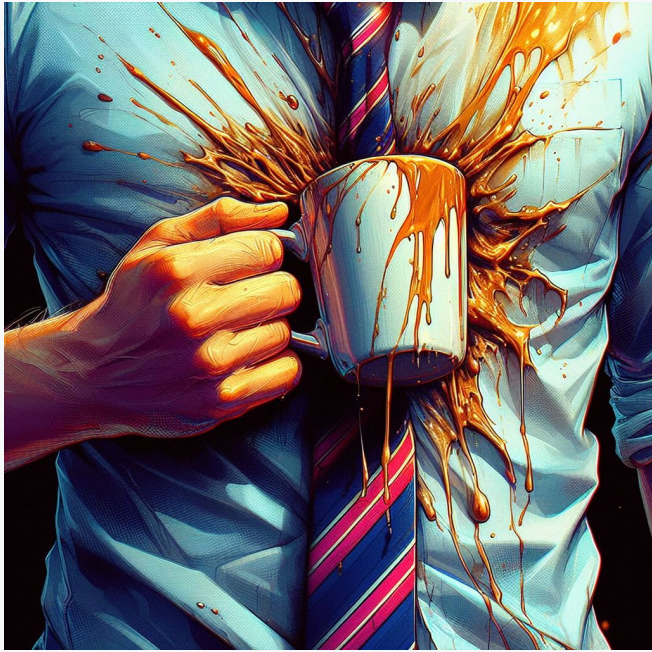


Image generation overview

Diffusion

Generating an image with **Diffusion Models** is like trying to remove the coffee stain from your shirt.

The model attempts to **reverse** the diffusion process, systematically **removing** the **noise** to recover the original clean data by predicting what was there before.

It's like having a magical stain remover that can perfectly clean your shirt, **restoring** it to its original state. If desired, it could even **redesign** the pattern on the shirt into something new, instead of just restoring it.

This reversal from **chaos** back to **clarity** is the essence of what makes diffusion models in AI so intriguing, since it can **extrapolate** from the original data and **create** completely new things by combining several pieces of other references.

Image generation overview

Diffusion

The same way that a GPT model needs **training** to work, the image generation starts with a huge database of **labeled data**, where you have the name of the elements and objects mapped correctly.

For example, these images:

two cats in a couch | mountains in a beach

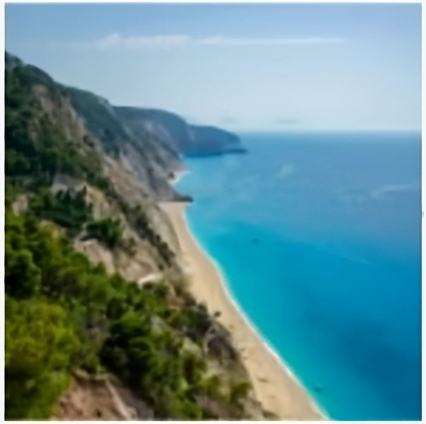
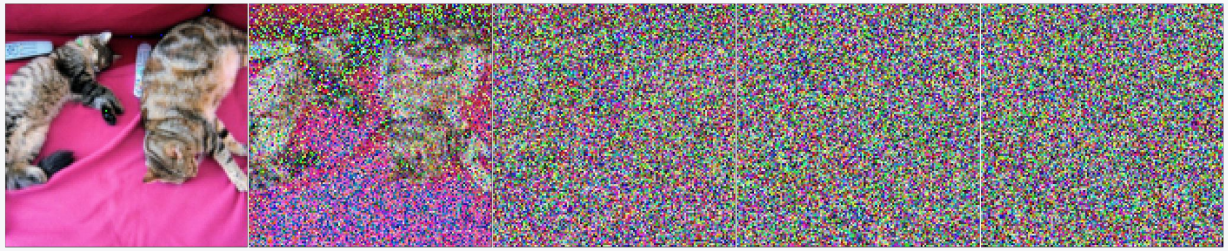


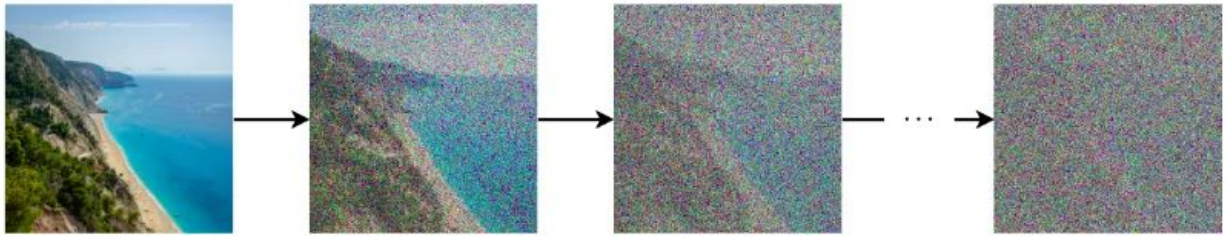
Image generation overview

Diffusion

Then the *forward diffusion* process turns the picture in a completely unintelligible image by adding noise. Since the data is **labeled**, you can train the model to relate these **unintelligible pixels** to the **original content** in the starting image.



Two cats in a couch



Mountains in a beach

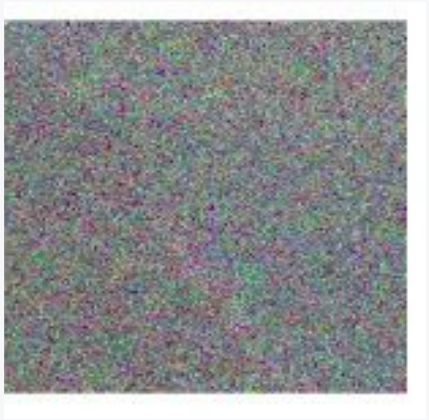
Image generation overview

Diffusion

Now, with the same techniques of **reinforced learning** as we saw before, the model can relate one of these images to cats, and the other to mountains in a beach.

This can be repeated **billions of times** (or more) until the model has information about pretty much any object or element that could be in your prompts in the future.

Two cats in a couch



Mountains in a beach

Image generation overview

Diffusion

When generating completely new pictures, you can pick a totally **random pixelated** image and use a *backward diffusion* process for the model to **infer** the most probable pixels and features that relate to the prompt provided.

Once the process is finished, the model will output a **slightly less** noisy image. To get a fully usable image, the model needs to run a few times in **loop** (doing sequential passes), until a clear image is outputted matching the prompt provided.

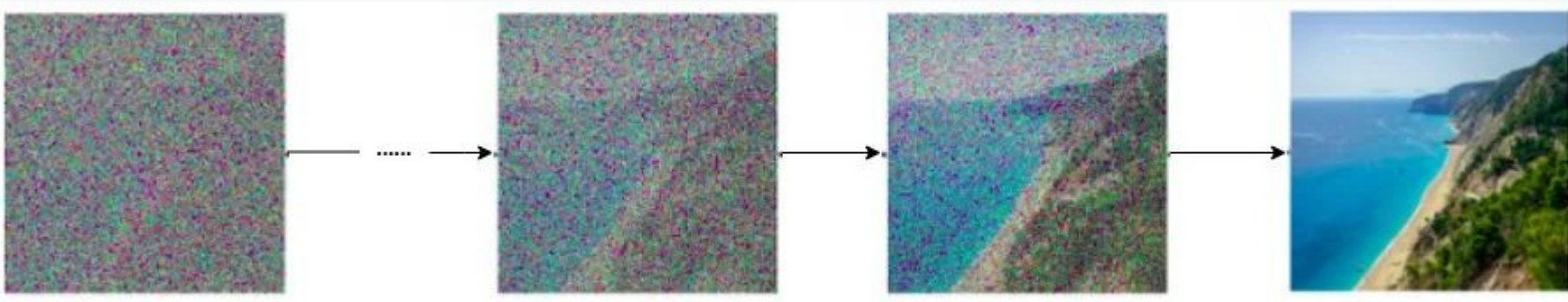


Image generation overview

Stable diffusion

There are many different diffusion models for generating images, like Denoising Diffusion Probabilistic Models (**DDPM**s), **Stable Diffusion**, Super-Resolution via Repeated Refinement (**SR3**), Guided Diffusion Models (**GDM**s), Cascaded Diffusion Models (**CDM**s), and many others.

All of these diffusion models use some sort of **Gaussian noise** to encode images from the training dataset. Then, they use some **noise predictor** algorithm together with a **reverse diffusion process** to recreate the image.

Stable Diffusion differs from many other models since it's **accessible** and **easy to use**. It is one of the first effective models that can run on consumer-grade graphics cards.

Image generation overview

Stable diffusion

Stable Diffusion doesn’t use the whole pixel space of the image. Instead, it uses a **reduced-definition latent space**.

The reason for this is that a color image with 512x512 resolution has 786,432 possible values. By comparison, Stable Diffusion uses a compressed image that is **48 times smaller** at 16,384 values, significantly reducing the amount of resources needed to run the model. The smaller latent space works because natural images aren't *random*.

This is why you can possibly use Stable Diffusion on a *normal* desktop with an NVIDIA GPU with 8Gb of VRAM.

Stable Diffusion uses Variational Autoencoder (VAE) files in the decoder to **paint fine details** like eyes.

Image generation overview

Sampling

To produce an image, Stable Diffusion first generates a **completely random** image in the latent space. The **noise predictor** then estimates the noise of the initial image, and then the predicted noise is subtracted from it. This process is repeated a many times to eventually output a clean image.

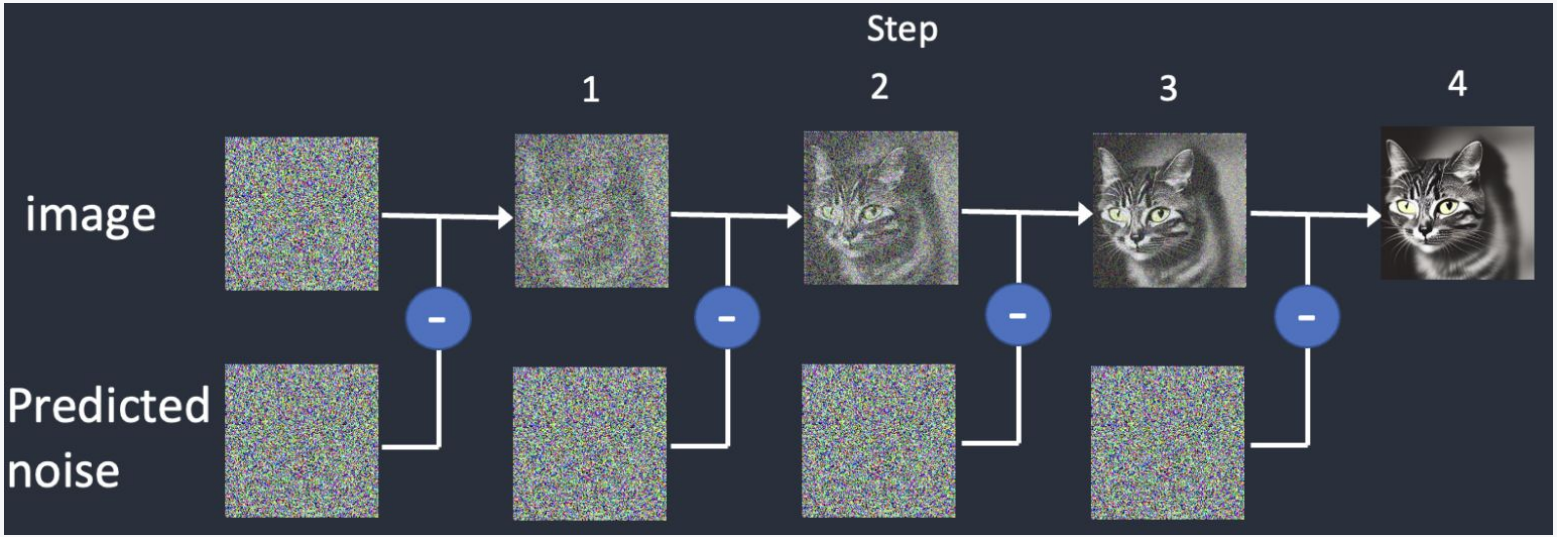


Image generation overview

Sampling

There are many different sampling methods that can be used to generate images.

These methods determine how the model **traverses the space** between pure noise and a coherent image.

VENTURE MINER

Image generation overview

Sampling

For example, these are some of the most common **denoising methods** applied to Stable Diffusion:

Euler-Maruyama Sampling is a basic approach to sample from the diffusion process. It's a form of **stochastic differential equation solver** and is used for its **simplicity** and **efficiency**. However, it may not always produce the highest quality results due to its straightforward approach.

Ancestral sampling is a more advanced approach where each step in the reverse diffusion process is **sampled** based on the **output** of the **previous step**, like a **chain**. This method can be more precise as it carefully builds on the previously generated data, leading to more **coherent** outputs.

DDIM (Denoising Diffusion Implicit Models) Sampling is a **non-Markovian** variant of the diffusion process that allows for **deterministic sampling**. It can generate images **faster** and sometimes with relatively **better quality**. It works by **implicitly denoising** the image in each step, hence the name.

VENTURE MINER

Image generation overview

Image generation models

GANs (Generative Adversarial Networks) consist of **two neural networks** which work in opposition to each other. One creates images (**generator**), and the other evaluates them (**discriminator**). Over time, the generator learns to produce more **realistic images**. Unlike Stable Diffusion, GANs typically don't use textual input but learn to generate images that are indistinguishable from real ones.

VAEs (Variational Autoencoders) work by **encoding** data into a **compressed image** and then **decoding** it back to the original form. They're effective in generating data that's **similar** to the training data. VAEs generally are not as **sharp** or **detailed** in their output as models like Stable Diffusion. They're better suited for tasks where an **approximation** of the data is sufficient.

DALL-E uses the **GPT** language models adapted for image generation. It generates outputs from textual descriptions, similar to Stable Diffusion. It uses the encoded text as a guide to generate a corresponding images by running a **modified version** of the **GPT's transformer architecture** trained on a dataset of **text-image pairs**. The text tokens in the prompts are evaluated with the pixels and features related to it in the training data to infer a most probable output image.

Hands-on experimentation

Setting up the environment

- Using a checkpoint (Snapshot of Model State)
  - runwayml/stable-diffusion-v1-5
- Installing the dependencies
  - Create a folder for your project
  - Create a python virtual environment (https://docs.python.org/3/library/venv.html)
    - `python3 -m venv venv/`
    - `. venv/bin/activate`
  - Install dependencies
    - `pip install diffusers transformers accelerate omegaconf`
- Creating a Python script
  - Create a new file named generate.py
  - Open the file in Visual Studio

VENTURE MINER

Hands-on experimentation

Running a model with Python

- Sample code:

```
import torch
from diffusers import StableDiffusionPipeline

pipe = StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5",
torch_dtype=torch.float16)
pipe = pipe.to("cuda")

prompt = "a top photo of a pizza with the best toppings"
pipe.enable_attention_slicing()
image = pipe(prompt).images[0]
image.save("output.png")
```

Hands-on experimentation

Running a model with Python

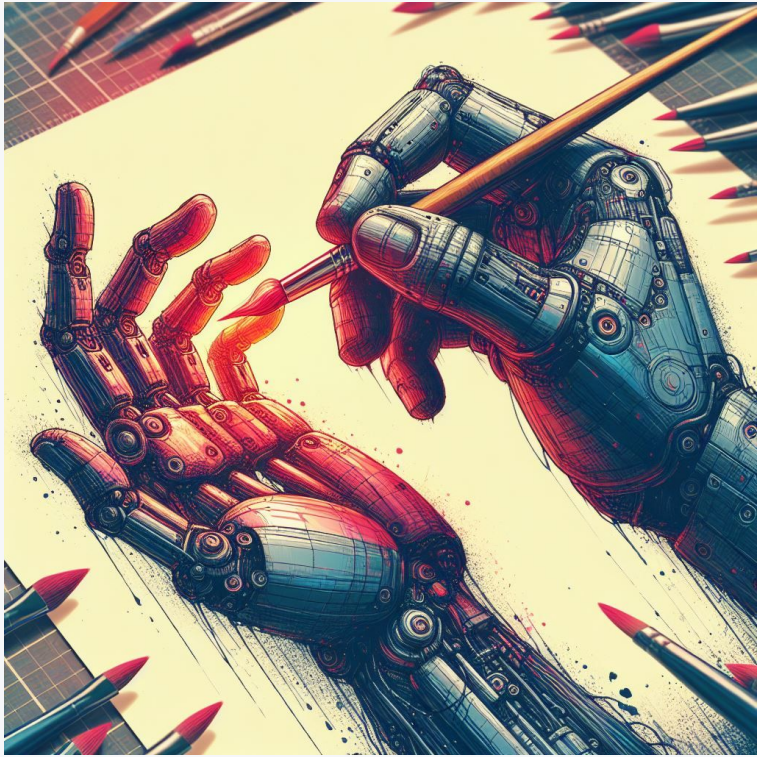
- Running the script
  - `python generate.py`
- Wait a few minutes for the model to load
- When the script finish running, open the *output.png* file



Running models with a local WebUI

Overview

- Stable Diffusion WebUI features
  - User-Friendly Interface
  - Customizable Parameters
  - Model picker
  - Extensions manager
  - Settings page
  - Community-Driven Features
  - Regular Updates and Improvements
- Repository
  - https://github.com/AUTOMATIC1111/stable-diffusion-webui
- Documentation
  - https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki



Running models with a local WebUI

Installing Stable Diffusion WebUI

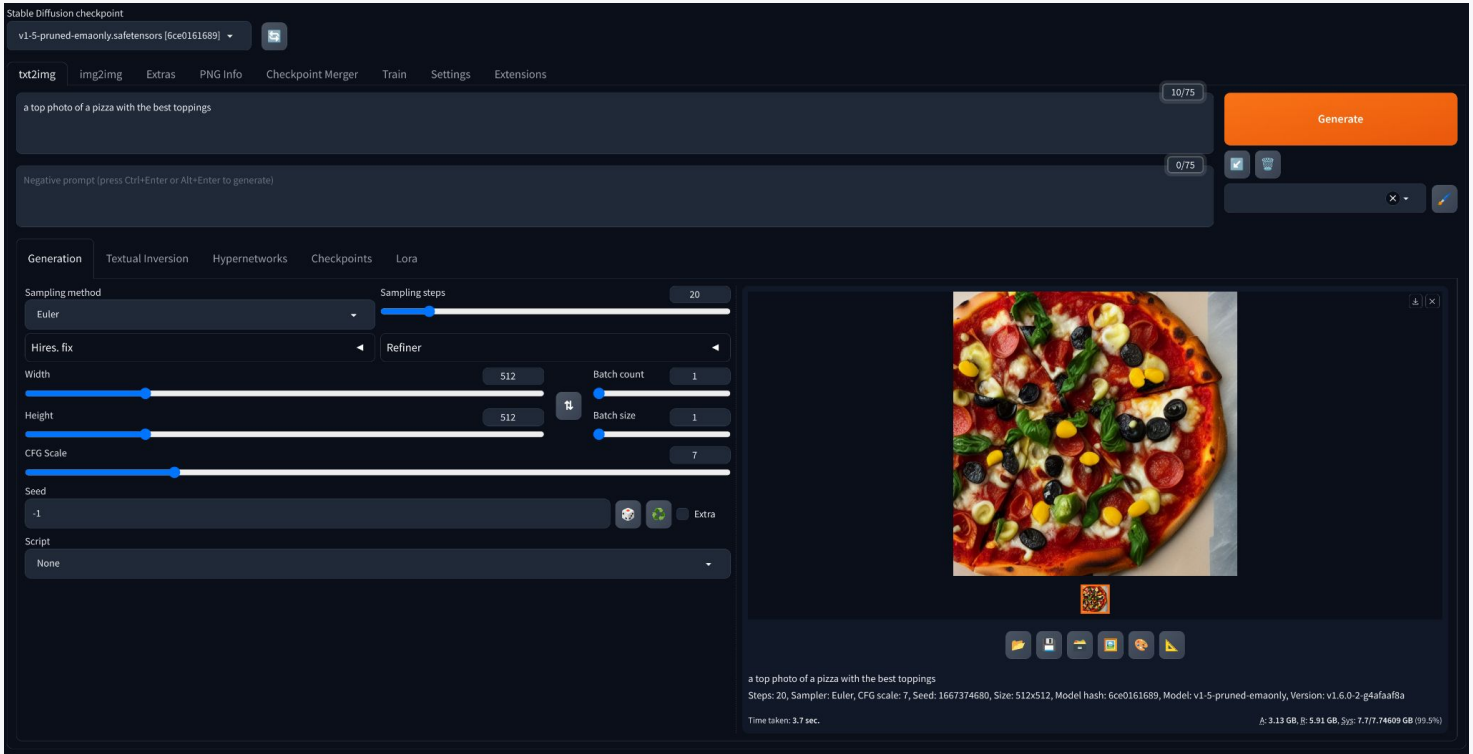
- Exit your virtual environment if it is still activated
  - `deactivate`
- Create a new folder for this project in a safe place
- Install all the dependencies https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki/Dependencies
- Install the package https://github.com/AUTOMATIC1111/stable-diffusion-webui#installation-and-running
  - Use `sudo chmod +x webui.sh` on linux if having permission problems
- Wait a bit for the dependencies to install and for the models to download
- Navigate to http://127.0.0.1:7860/ when the process is completed

- Alternatively you can install Stable Diffusion WebUI with Stability Matrix
  - Download the installer from https://lykos.ai/downloads
  - Open the app and go to the “Packages” tab
  - Click on “+ Add Package” and select “Stable Diffusion Web UI”

VENTURE MINER

Running models with a local WebUI

Using Stable Diffusion WebUI

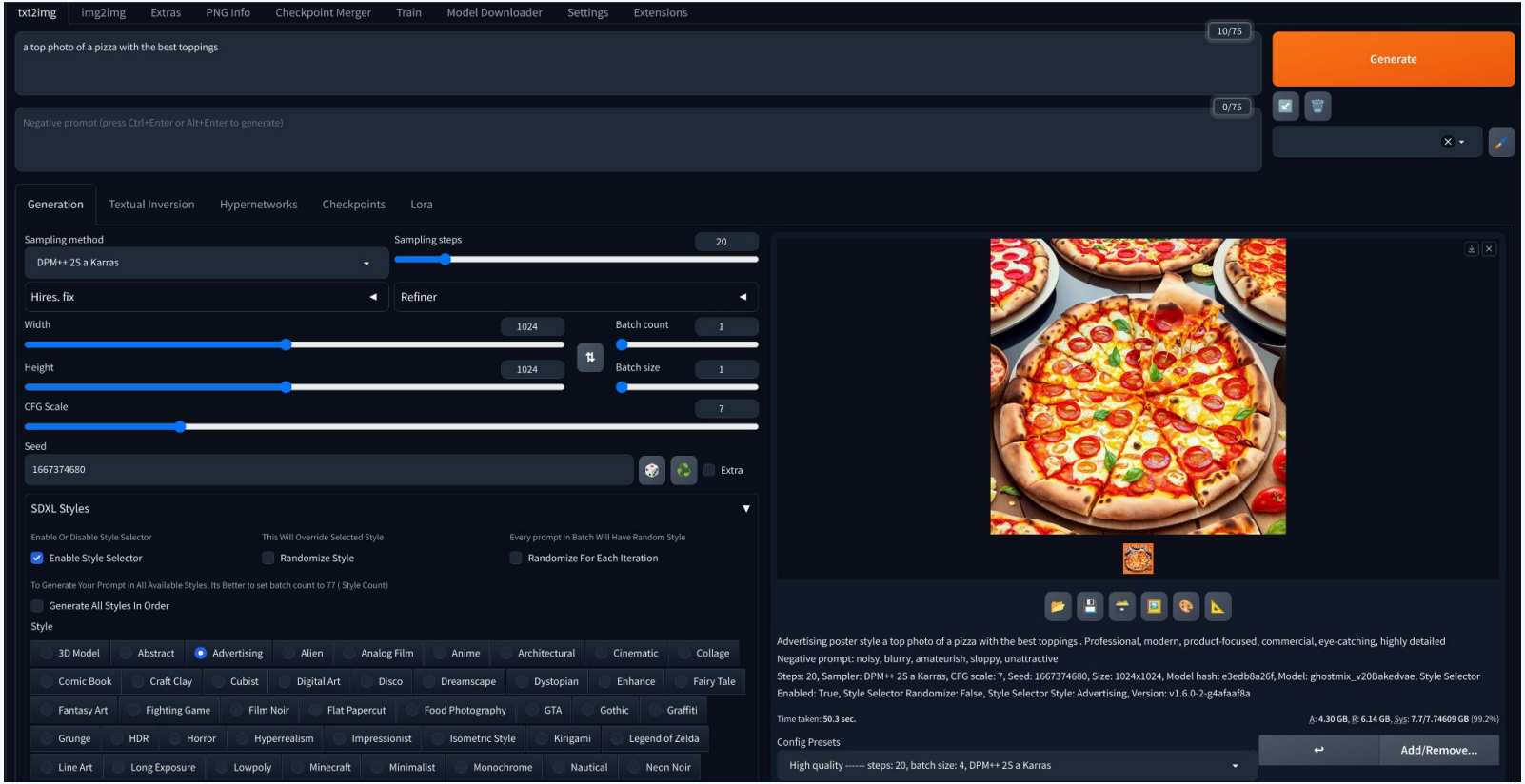


Running models with a local WebUI

Using Stable Diffusion WebUI

- Enter the stable-diffusion-webui folder
  - `cd stable-diffusion-webui`
- Run the WebUI
  - `./webui.sh` (Linux)
- Run the WebUI with the API
  - `./webui.sh --api`
  - You can view the API Swagger documentation at http://127.0.0.1:7860/docs
- Recommended: Install the “Model Downloader” extension
  - Extensions tab > Install from URL > URL for extension's git repository: https://github.com/Iyashinouta/sd-model-downloader
  - Download other models, like https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0/resolve/main/sd_xl_base_1.0_0.9vae.safetensors
  - https://civitai.com/api/download/models/76907
- Other recommended extensions: https://github.com/Zyin055/Config-Presets https://github.com/Mikubill/sd-webui-controlnet https://github.com/ahgsql/StyleSelectorXL

Experimentation



VENTURE MINER

Experimentation

Experimenting with Stable Diffusion WebUI

- Different checkpoints
  - https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0
  - https://civitai.com/api/download/models/76907
  - https://civitai.com/models/84728/photon
- Sampling methods
  - Prediction vs randomness
  - Convergence
  - Speed
- Sampling steps and Batch sizes
- CFG Scales
- Prompts
  - Positive prompts
    - Elements and keywords
  - Negative prompts
- Resolutions

Experimentation

Sampling methods recommendations

- Old-School ODE solvers
  - Euler – The simplest possible solver
  - Heun – A more accurate but slower version of Euler
  - LMS (Linear multi-step method) – Same speed as Euler but (supposedly) more accurate
- Ancestral samplers
  - Euler a, DPM2 a, DPM++ 2S a and DPM++ 2S a Karras – Stochastic samplers that adds noise to the image at each sampling step, adding some randomness to the final image
- Karras noise schedule
  - Many samplers with the label “Karras” – Smaller noise step sizes to improve quality
- DDIM and PLMS – Somewhat outdated and not widely used anymore
- DPM, DPM2 and DPM++ – New samplers designed for diffusion models released in 2022
- UniPC – Sampler released this year, and may achieve quality image generation in 5-10 steps

- Comparison article: https://stable-diffusion-art.com/samplers/

Experimentation

Steps and CFG Scales

The amount of **sampling steps** can greatly influence the quality of the final image for most common sampling methods. Also, each extra step adds more processing time to generate the image.

The configuration for the Classifier Free Guidance (**CFG**) scale can control how closely the model should follow your text prompt during the sampling steps. Lower values (close to 1) allows for more freedom, while higher values (above 10) are more restrictive and can interfere with the quality.

Comparison article: https://www.artstation.com/blogs/kaddoura/pBPo/stable-diffusion-samplers

Experimentation

Prompts recommendations

- Positive prompts
  - Step 1: subject or content + action + form + mood
    - <cat/dog/person> <looking up/standing> <joyfully/boldly> <nostalgic/colorful>
  - Step 2: Image composition elements, like style and artistic form
    - <photography/painting/sculpture> <close-up shot/profile/landscape>
    - If the model was trained for it, you can add styles like expressionism or minimalist and even famous artists names or art niches recognized in the training data
  - Step 3 (optional): refinements like lighting, color scheme, detail level, realism and such
- Negative prompts
  - May filter bad results but also can limit the quality of the image generation process
  - Different subjects and styles require completely different negative prompts
    - Living creatures -> deformed, ugly, too many fingers, too many limbs
    - Objects -> duplicate, blurry, pixelated, low quality, out of frame, cropped
    - Faces -> poorly rendered face, deformed, disfigured, long neck, ugly
    - Landscape -> text, watermark, low quality, poorly drawn, pixelated
- More examples: https://github.com/Dalabad/stable-diffusion-prompt-templates

Conclusion

- Join our Telegram Group: https://t.me/vmineracademy
- Follow us on Twitter: https://twitter.com/ventureminer and https://twitter.com/encodeclub