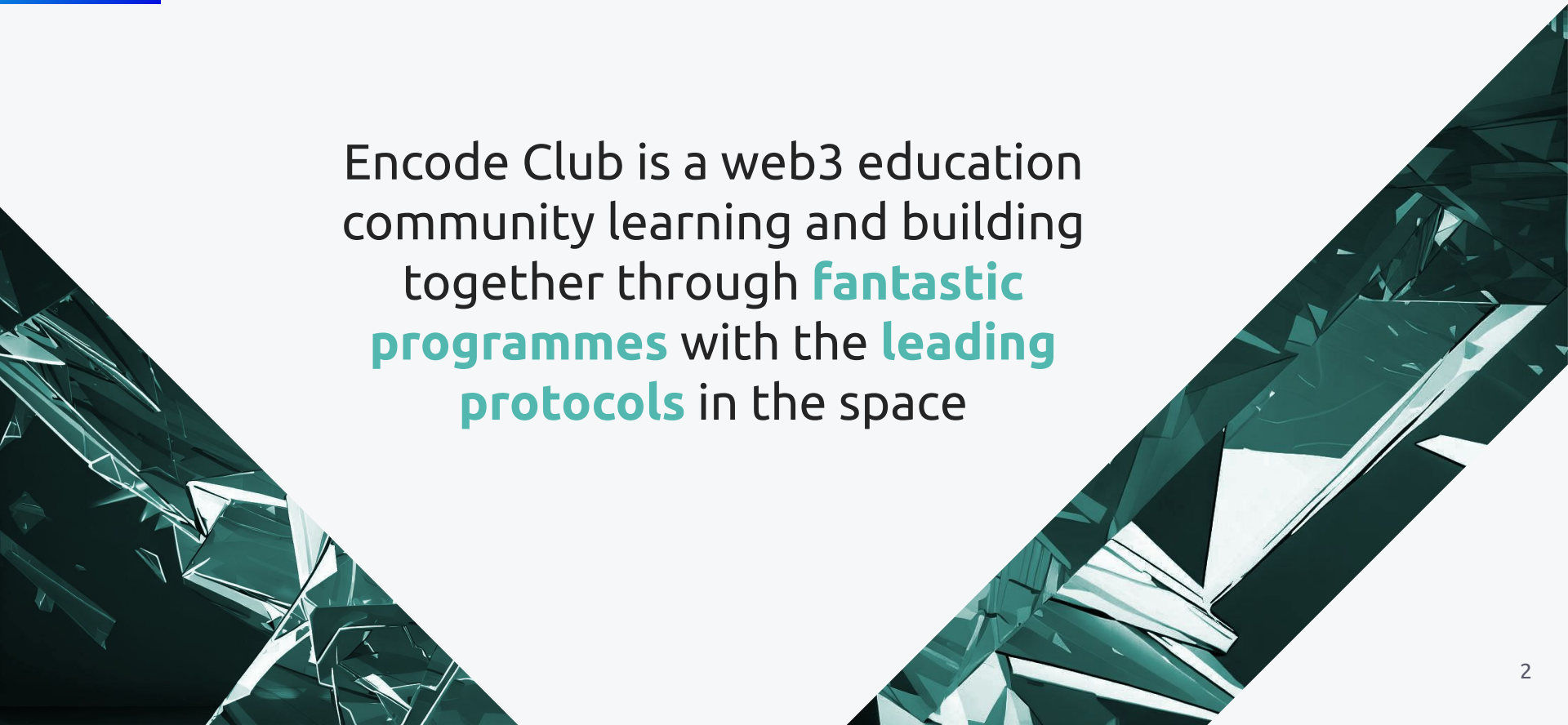
VENTURE MINER

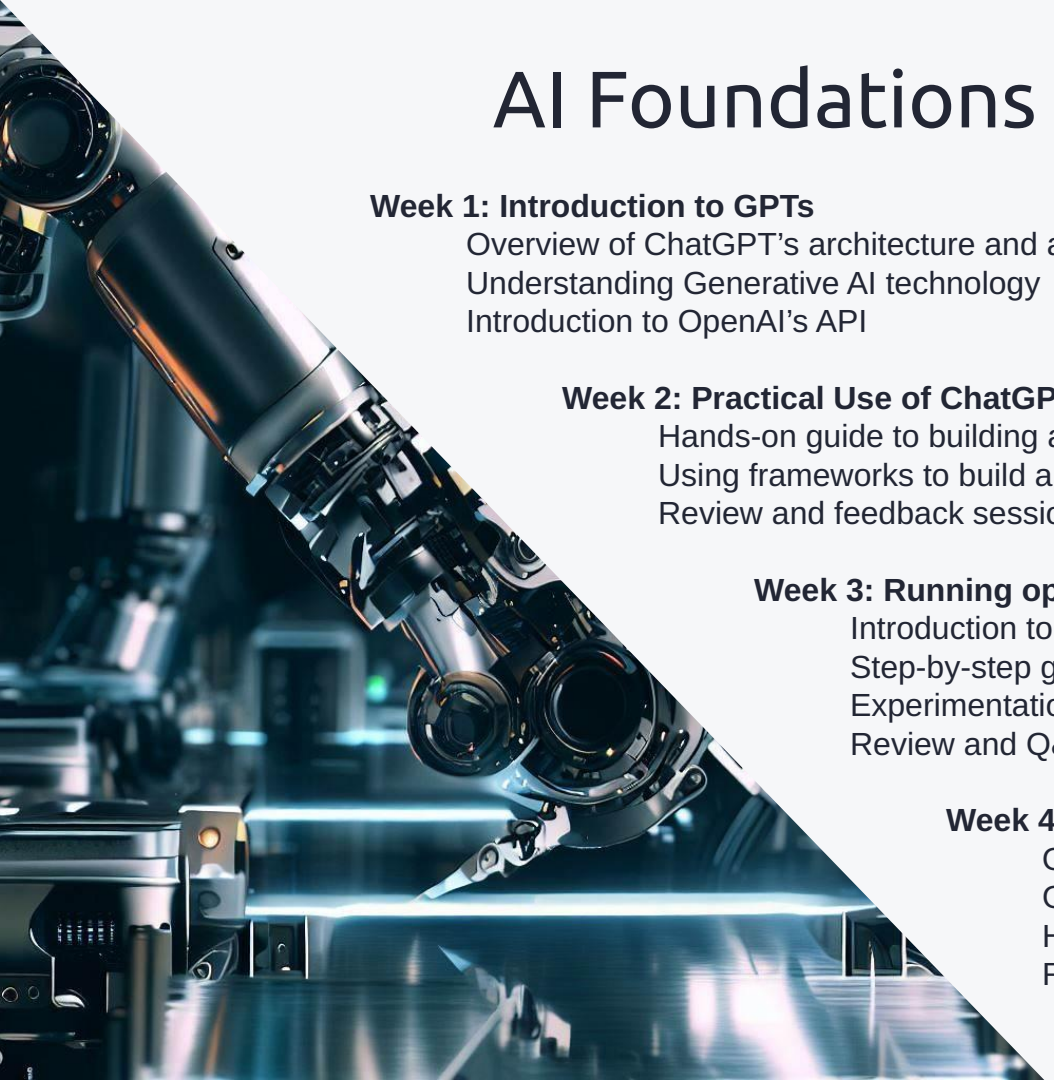
Early stage Web3 and AI
Venture Studio

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

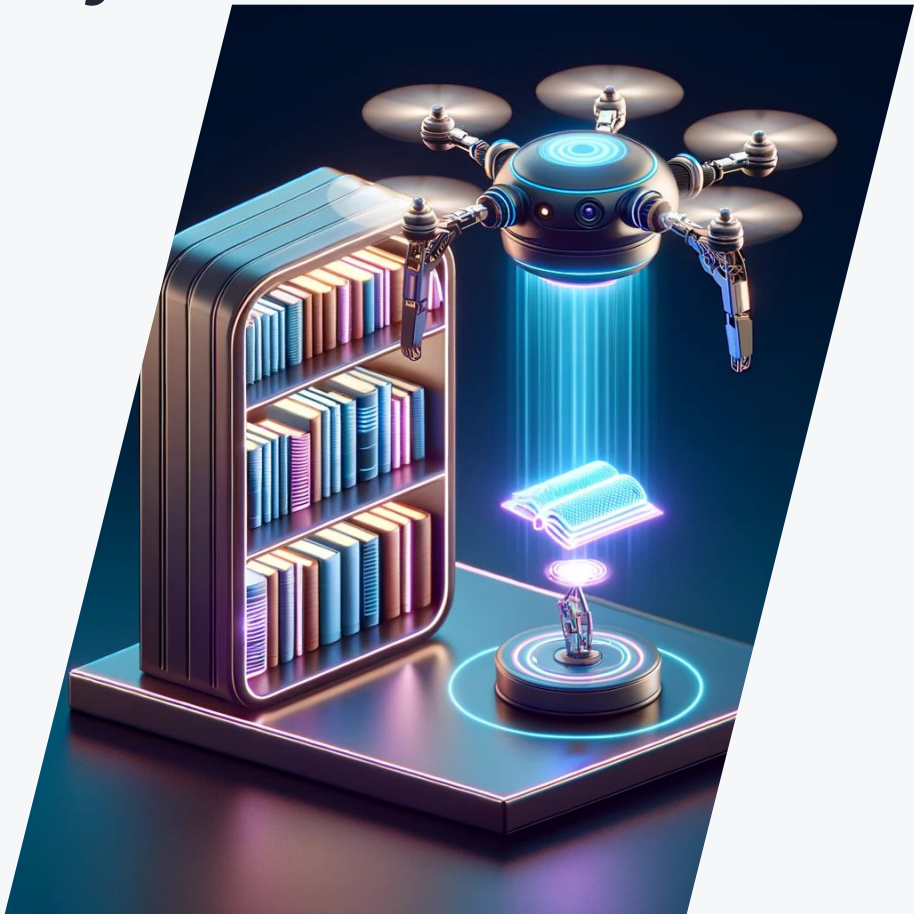
- Overview of image generation techniques
- Guide to executing scripts for image generation
- Hands-on experimentation with image generation
- Review and Q&A on image generation techniques

VENTURE MINER

Running open models locally

Topics

- ChatGPT Review
  - API Calls
  - Benefits
  - Risks
- Running a LLM model
  - Models overview
  - Picking a model
  - Using Hugging Face
  - Open Source models
- Using open models
  - Downloading models
  - Fine tuned versions
  - Modified versions
  - Running a model with python
- Running models with a local WebUI
  - Setting up
  - Environment configuration
  - Using Text Generation WebUI
- Experimentation
- Next steps



VENTURE MINER

Review

Using OpenAI APIs for building a Chat Application

- Using NLP models over API calls
  - Many benefits
    - Easy to implement
    - Easy to maintain
    - Flexible costs
  - Potential risks
    - Pricing models can change
    - Service outages and performance issues
    - Data management and data leaks
    - Censorship and bias
    - Vendor lock-in
- Using your own model
  - Open-source models
  - Running the models
  - Serving the model to applications
  - Managing servers

Why not to depend in a single company?

What could go wrong?

Review

Environment setup

- Create a *Hugging Face* account
  - https://huggingface.co/join
- Request access to LLAMA 3
  - https://ai.meta.com/llama/
    - Fill the form and click on [Download model]
    - Wait some hours or days for confirmation
    - Download all models (links expire in 24 hours)
- Prepare Python tools
  - *Python*, *Pip* and *Conda*
- (Optional alternative If python doesn't work) Prepare Docker tooling
- Setup Text Generation WebUi
  - https://github.com/oobabooga/text-generation-webui
  - Follow the installation instructions
  - Run the *start* script and navigate to the WebUI
  - Download some models in the *Models* tab (for example TheBloke/Llama-2-7B-Chat-GGUF)
    - View more on https://github.com/Troyanovsky/Local-LLM-Comparison-Colab-UI

Running a LLM

Overview

The LLMs running behind the OpenAI APIs are **Generative Pre-trained Transformers** (GPTs).

These models are a family of neural networks that uses a **transformer** architecture to make **Inferences:**

- Given some fragments of letters, syllables or even words (the *n-grams*), the transformer **deep learning** architecture can compute (*infer*) what other fragments would be more probable to be found together with the ones provided as input

This architecture requires the model to be **trained** into specific sets of data, in order for these fragments to be properly related to each other inside the model.

Running a LLM

Overview

Alone, these transformers wouldn't generate helpful answers to prompts in natural language. These models need to be **Fine Tuned** to give helpful answers to questions and tasks.

These fine tuned models could be adapted to answer to specific formats, like common **chat**, or **code** completion or even **specific tasks** like composing cooking recipes or researching chemicals substances.

All of this **training** and **tuning** requires a tremendous amount of **resources** to gather and compute all the information needed, so it's quite normal for companies and organizations to seeks some sort of **financial compensation** for putting out the work into building these models.

This is why **Open Source** LLM models are of such importance.

Running a LLM

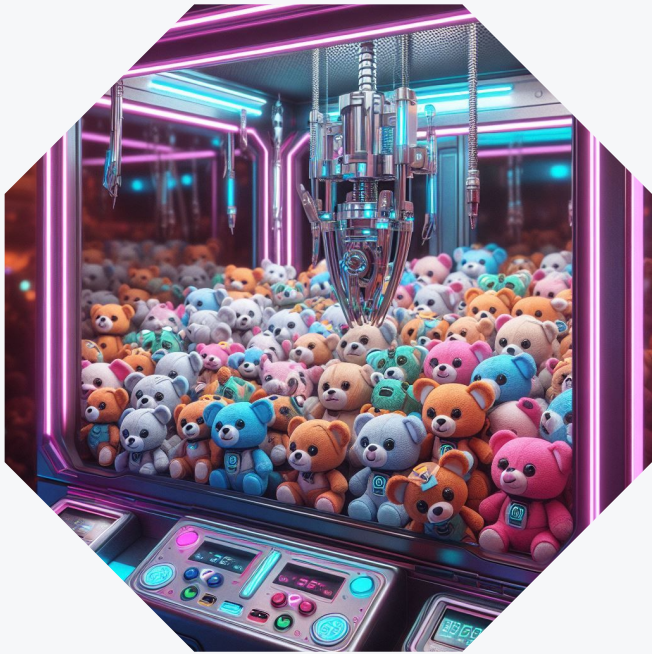
Picking a model

When using an API for running the model for your application, you only have a **few options** available given by the API service provider itself.

When running **your own model** for your application, you're now able to use any kind of compatible model you manage to get access to.

The best option for a quick (and affordable) start would be to download an **open mode**l and running it in your **own device**.

For the application to work properly, you would need to consider the **recommended requirements** for each model and the **proper tools** to correctly execute the prompts against the models.



Running a LLM

Picking a model

A great way to find (and even implement) models is to use *HuggingFace* (aka 🤗) and other similar hubs and communities.

- Hugging Face Inc. is the American company that created the Hugging Face platform. The company funded in 2016 originally developed a chatbot app by the same name for teenagers. The company switched its focus to being a machine learning platform after open sourcing the model behind the chatbot app.

You can use this application to **find** and share **models** and **data sets**, collaborate on researches and even fine-tune and train deep learning models.

VENTURE MINER

Running a LLM

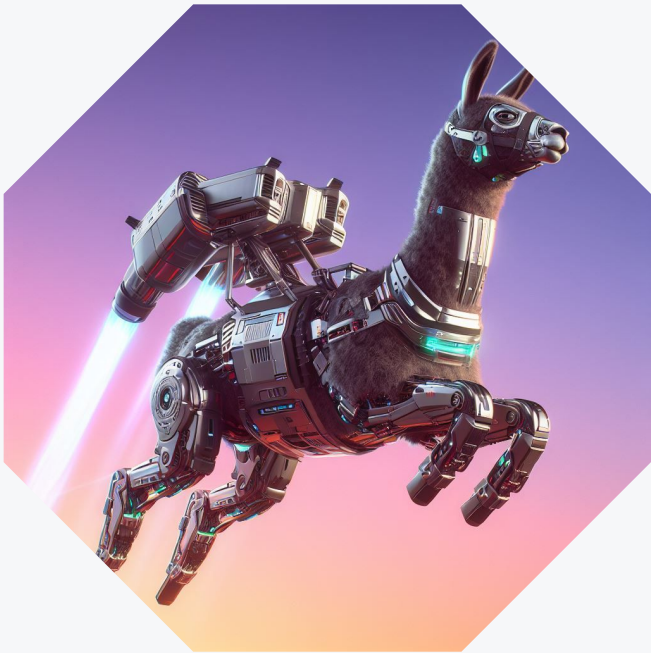
LLaMA

Some of the most robusts *source-available* models that we can openly access at the moment are in the **LLaMA** family, released by **Meta** earlier 2023.

The most recent **LLaMA** (Large Language Model Meta AI) models are available to download **free of charge** for research and commercial use by filling a request form in their website.

Meta trained and released **LLaMA 3** on April 18th in **8B** and **70B** parameters sizer, and these includes both foundational models and fine-tuned versions for instructions, called **LLaMA 3 Instruct**.

Since Meta made the model *(almost***)* open source, there are several **community maintained variations** and adaptations published for free in many forums and AI hubs.



VENTURE MINER

Running local models

Choosing the models

- Hugging Face model
  - https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
- Comparing models
  - https://github.com/Troyanovsky/Local-LLM-Comparison-Colab-UI
- Top choices
  - https://huggingface.co/TheBloke/Mistral-7B-OpenOrca-GGUF
  - https://huggingface.co/TheBloke/Llama-2-13B-chat-GGUF
  - https://huggingface.co/TheBloke/wizard-vicuna-13B-GGUF
- Lightweight choices
  - https://huggingface.co/TheBloke/Llama-2-7B-Chat-GPTQ
  - https://huggingface.co/TheBloke/Mistral-7B-OpenOrca-GGUF/blob/main/mistral-7b-openorca.Q2_K.gguf
  - https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF/blob/main/llama-2-7b-chat.Q2_K.gguf

Running local models

Running the models

- Most used clients
  - Llama.cpp
    - https://github.com/ggerganov/llama.cpp
  - Text Generation WebUI
    - https://github.com/oobabooga/text-generation-webui
  - Kobold cpp
    - https://github.com/LostRuins/koboldcpp
  - Llama cpp python
    - https://github.com/abetlen/llama-cpp-python
- Usage alternatives
  - Local install
  - Docker images
  - Cloud instances
  - Virtual workspace/laboratory

Running local models

Running a simple model with python transformers

- Using an adapted version of LLaMA 2
  - TinyLlama/TinyLlama-1.1B-Chat-v1.0
- Installing the dependencies
  - Install PyTorch in your device (https://pytorch.org/get-started/locally/)
  - Create a folder for your project
  - Create a python virtual environment (https://docs.python.org/3/library/venv.html)
    - `python -m venv venv/`
      - You can use `python3` or `py` if your system doesn't recognize `python` command
    - `. venv/bin/activate`
  - Install dependencies
    - `pip3 install transformers optimum`
- Creating a Python script
  - Create a new file named run.py
  - Open the file in Visual Studio or any IDE or Text Editor of your choice

VENTURE MINER

Running local models

Running a simple model with python transformers

- Sample code:

```
import torch
from transformers import pipeline

model_name = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"
pipe = pipeline("text-generation", model=model_name)

messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot named Joe who always responds with witty comments"
    },
    {"role": "user", "content": input("Say something...\n\n")},
]

print("Generating answer...\n\n")

prompt = pipe.tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)
outputs = pipe(prompt, max_new_tokens=64, do_sample=True, temperature=0.7, top_k=50, top_p=0.95)
print(outputs[0]["generated_text"])
```

Running local models

Running a simple model with python transformers

- Running the script
  - `python run.py`
- Wait a few minutes for the model to load
- Output example (after download)

```
Say something...

Hello. What is your name?
Generating answer...


<|system|>
You are a friendly chatbot named Joe who always responds with witty comments</s>
<|user|>
Hello. What is your name?</s>
<|assistant|>
I do not have a name like humans. However, you can call me joseph, if you prefer.
```

Running models with a local WebUI

Overview

- Text Generation WebUI features
  - Simple user interface for **downloading**, **configuring** and **running** many different models
  - Many Plugins and Extensions
  - Three usage modes
    - Chat
    - Two columns
    - Notebook
  - Multiple model running clients
  - Quick load, unload and switch between models
  - OpenAI-compatible API server
- Repository
  - https://github.com/oobabooga/text-generation-webui
- Documentation
  - https://github.com/oobabooga/text-generation-webui/wiki

Running models with a local WebUI

Installing Text Generation WebUI

- Exit your virtual environment if it is still activated
  - `deactivate`
- Create a new folder for this project in a safe place
- Clone the repository
  - `git clone https://github.com/oobabooga/text-generation-webui.git`
- Enter the folder
  - `cd text-generation-webui /`
- Run the installation script depending on your OS
  - `start_linux.sh` on Linux
  - `start_windows.bat` on Windows
  - `start_macos.sh` on MacOS
  - `start_wsl.bat` on WSL
- Pick your GPU option
  - You can pick “None” (N) to use models in CPU mode only if needed
- Wait a few minutes for the installation to finish

Running models with a local WebUI

Using the Text Generation WebUI

- Activate a virtual environment with conda
  - `conda activate installer_files/env`
- Run the server
  - `python server.py`
    - You can set a custom path to your directory with all the models
    - `python server.py --model-dir ../path/to/your/models/folder`
- Navigate to your local URL
  - http://127.0.0.1:7860
- Choose a model in the “Models” tab
- Click on “Load”
- Try out the chat

Alternatively, if it fails, try running it with Docker

- https://github.com/oobabooga/text-generation-webui/wiki/09-%E2%80%90-Docker

Running models with a local WebUI

Running a local API server

The server script can also provide a direct drop-in replacement to the OpenAI API

- Activate a virtual environment with conda
  - `conda activate installer_files/env`
- Run the server with the API flags
  - `python server.py --api --listen`
- Navigate to your local URL
  - http://127.0.0.1:7860
- Choose a model in the “Models” tab
- Click on “Load”
- Try out the API with swagger
  - http://127.0.0.1:5000/docs

Running models with a local WebUI

Running a local API server

- Try out the API with *curl*

```
curl http://127.0.0.1:5000/v1/completions \
 -H "Content-Type: application/json"  \
 -d '{"prompt": "This is a cake recipe:\n\n1.", "max_tokens": 200, "temperature": 1,
"top_p": 0.9, "seed": 10 }'
```

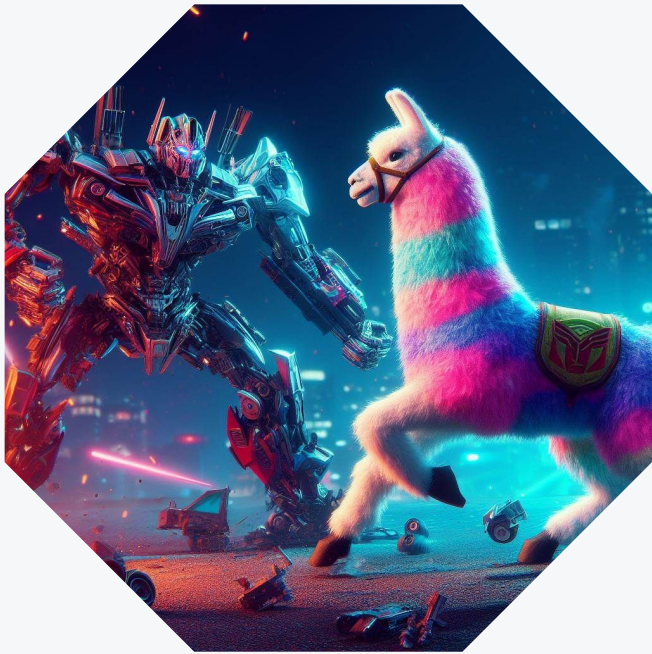
- Output

```
1. Preheat the oven to 350°F.Grease and flour a 9x13-inch baking pan.
2. In a large mixing bowl, cream together the butter and sugar until light and fluffy.
Beat in the eggs one at a time, followed by the vanilla extract.
3. In a separate bowl, whisk together the flour, baking powder, and salt. Gradually
add the dry ingredients to the wet ingredients and mix until just combined.
4. Pour the batter into the prepared pan and smooth the top.
5. Bake for 30-35 minutes or until a toothpick inserted into the center comes out
clean.
6. Remove the cake from the oven and let it cool completely in the pan before frosting
and serving.
```

Experimentation

Putting the models to use

- Try using Google Colab notebooks
  - https://colab.research.google.com/github/oobabooga/text-generation-webui/blob/main/Colab-TextGen-GPU.ipynb
  - https://github.com/camenduru/text-generation-webui-colab
- Try running inside your own cloud instance
- Try extensions like Character Bias, Sending Pictures, Multimodal, Google Translate and others
  - https://github.com/oobabooga/text-generation-webui/wiki/07-%E2%80%90-Extensions
- Try downloading extensions from the community
- Try attaching your frontend to your own API
- Try creating a quick product demo and raising millions of dollars from hasty VCs dying of FOMO
  - Please, don't do it (again)! I'm just kidding.



Next steps

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

VENTURE MINER

Thank you!