Purpose Statement

The current gold standard for AI image generators is the OpenAI open-source project called DALL-E 2. While the current model of DALL-E is fantastic at making beautiful artwork like the one below it also has some severe limitations in image generation that we aim to take a stab at.

*“a color pencil drawing*

*of a ship at sea at*

*dusk”*

One problem with DALL-E’s image generation is that when asked to generate an image with text the resulting text is a jumbled mess of letters. Pictured below is a sample query for *alternative Nike logos*. Out of the 13 photos with words, only one image has “Nike” spelled correctly.

****

For our model, we want to create an image generator that can produce company logos in a professional and legible manner. This project would extend off of DALL-E’s open-source code to improve the text-in-image generation for use in logos.

Problem Statement

Logos are a major part of a brand’s identity and designing them is an incredibly difficult task for any graphic designer. The design must reflect the spirit of the company while accounting for the specific requirements that a non-artistic executive is demanding. The perfect logo is a combination of artistic vision and company image/values.

A well-designed logo can make a company or a brand stand out from its competitors. There is no better example of this in action than the Nike Swoosh. The Swoosh was designed by a student in 1971 for 35$ and has set the standard for what all logos should look like and the purpose that they serve. What makes the Swoosh so special is that it's so incredibly simple and yet invokes such a strong sense of speed and athleticism which is what Nike is all about.

In recent years there has been an effort to automate the design process of a logo using “artificial intelligence.” These applications are often lacking in depth of design and really any usage of true artificial intelligence. This is where our project hopes to combine the artistic vision of graphic designers with artificial intelligence to generate visually pleasing logos.

Links:

Code

<https://github.com/lucidrains/DALLE2-pytorch/blob/main/README.md>

Repo for x-clip <https://github.com/lucidrains/x-clip>

<https://www.geeksforgeeks.org/converting-an-image-to-a-torch-tensor-in-python/>

<https://github.com/openai/CLIP/blob/main/notebooks/Interacting_with_CLIP.ipynb> -awesome intro notebook

<https://www.reddit.com/r/MachineLearning/comments/xslpwt/p_pok%C3%A9mon_text_to_image_fine_tuned_stable/?utm_source=share&utm_medium=ios_app&utm_name=iossmf>

<https://colab.research.google.com/github/AK391/lambda-diffusers/blob/main/notebooks/pokemon_demo.ipynb#scrollTo=IebtTUPpoORj> - very similar project to what we are doing

<https://github.com/justinpinkney/stable-diffusion/blob/main/examples/prior_2_sd.ipynb> - pytorch image generation

<https://huggingface.co/datasets/lambdalabs/pokemon-blip-captions>

<https://lambdalabs.com/blog/how-to-fine-tune-stable-diffusion-how-we-made-the-text-to-pokemon-model-at-lambda/>

<https://github.com/marqo-ai/marqo/blob/mainline/examples/StableDiffusion/hot-dog-100k.md>

pip install torch==1.11.0+cu113 torchvision==0.12.0+cu113 torchaudio==0.11.0 --extra-index-url <https://download.pytorch.org/whl/cu113>

pip install ipywidgets

**The first milestone is to get image diffusion to work locally.**

To do this we imported an image dataset from Kaggle hosted by Microsoft. [Link](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbXN3SjFXcVkwZjBxVEhvQ2REOHZQQ1Y5b2NPZ3xBQ3Jtc0ttTVlWYnJVZ0NCU1dRNEs1RmtxT1JMQkFETmhuTS1sRnZUenBXaXFrTXU3M1BhT3VXTTQwbjFmejY4NE41S1hsMmdfbjVVaFpIU0JsSU1NY0RUSHExUHIyd2JGeF9ULVB6S3U5NXgxZzlwZEhOSEpTdw&q=https%3A%2F%2Fwww.microsoft.com%2Fen-us%2Fdownload%2Fconfirmation.aspx%3Fid%3D54765&v=j-3vuBynnOE) [Tutorial for data](https://www.youtube.com/watch?v=j-3vuBynnOE)

The first goal was to get the model to be able to draw a cat given a bunch of images of just cats.

The first significant issue ran into. CLIP wants a 4, 3, 256,256 but the best I can do is 1,3,256,256.

Fix: It turns out that it was just the input dimension of the test text so I just worked around it.

The next issue is figuring out how to input more than one data point and one iteration of the model so that it can produce actual images.