

CS410 Final Project Report


Team: VisualTopics

Team Members: Jason Lundquist (captain), Molly Yang, Louie Hernandez

Team Members NetID: jasondl3, ty2, lgherna2

Visual Topics Site: <https://cs410-sd-app.github.io/>

Video: [CS410 Final Project - Visual Topics.mp4](#)

Slides:  CS410 Final Project Presentation

Code Repo: https://github.com/mollytyy/stable_diffusion.git

State-of-the-art text to visualization applications exist today that bring together information retrieval and deep learning to create stunning images from merely a couple of words in the query. The models are trained on billions of text image pairs.¹ However, this extraordinary ability comes with certain limitations and societal impact. Some examples include the datasets (LAION) reflecting harmful stereotypes, oppressive viewpoints, and social bias. Efforts were made to filter out inappropriate content and toxic language; however, some like Imagen decided not to release the software for public use considering the risk of the model encoded harmful representations.²

Our goal for this project was to explore a route to not only exclude the harmful content, but also apply this state-of-the-art technology to education. Many of the results we get from today's text to image models are utilized for a quick and fun way to conceptualize an idea, entertainment, or mimicking art styles. We wondered if the same technology could contribute to students and educators in classrooms. When it comes to studying for subjects like math, physics, or chemistry, having an image or diagram along a long paragraph of text could make a world of a difference. Like having subtitles to a video, we want to be able to perform the reversed task and come up with a visualization (image, slides of images, or even a video) to a paragraph. In this report, we outline our results relative to our project goals and review the instructions on how to run our application and a high level overview of our code. Our project goal was mostly realized and we are satisfied with the results and hope that you find value in our work.

To start, visit our website at cs410-sd-app.github.io. It should bring you to a page that looks like this. Titled "ESL Text to Image - Instant Search" and a box for a paragraph. (Open website) Enter a paragraph and click on search for our application to start generating images. Here is an example from the prompt the site generated:

¹ <https://laion.ai/blog/laion-5b/>

² <https://imagen.research.google/>



This was one of the images generated from the follow paragraph text:

"Monday was the day of the big test. Luella had three days left to study. She knew she had to study for at least three hours per day in the next couple of days if she hoped to get the grade she wanted. Luella needed to score at least a ninety percent in her test in order to pass her Algebra class with an A."

To produce an image like the one above, navigate to our hosted site and click on "Generate Prompt" then "Search" for the visualizations to load. Images are generated corresponding to each sentence entered into the prompt. Note that the results could be different each run due to how the model generates images. Along with interacting with our website, there are some links to repos and colab notebook if you want to explore further on this topic, please find these on the title page just under the title of this report.

Code Function

Stable Diffusion is a latent text-to-image diffusion model that is trained on an Latent Diffusion Model on 512x512 images from a subset of the LAION-5B database. Similar to Google's Imagen, this model uses a frozen CLIP ViT-L/14 text encoder to condition the model on text prompts. Stable Diffusion v1 refers to a specific configuration of the model architecture that uses a downsampling-factor 8 autoencoder with an 860M UNet and CLIP ViT-L/14 text encoder for the diffusion model. The model was pre-trained on 256x256 images and then fine tuned on 512x512 images. Stable Diffusion v1 is a general text-to-image diffusion model and therefore mirrors biases and (mis-)conceptions that are present in its training data. The weights are available via the CompVis organization at Hugging Face under a license which contains specific use-based restrictions to prevent misuse and harm as informed by the model card, but otherwise remains permissive. The CreativeML OpenRAIL M license is an Open RAIL M license, adapted from the work that BigScience and the RAIL Initiative are jointly carrying in the area of responsible AI licensing. As a stretch goal, we endeavored to customize further where a student could query a basic electronic circuit, and the model would generate a circuit representation. Stable diffusion outputs did not represent circuit diagrams clearly as we would want for educational purposes. We attempted to use textual inversion to train the model with images about electronics to produce a custom checkpoint; however, we ran into pytorch tensor error when running locally, and the cloud instances require greater compute resources than we

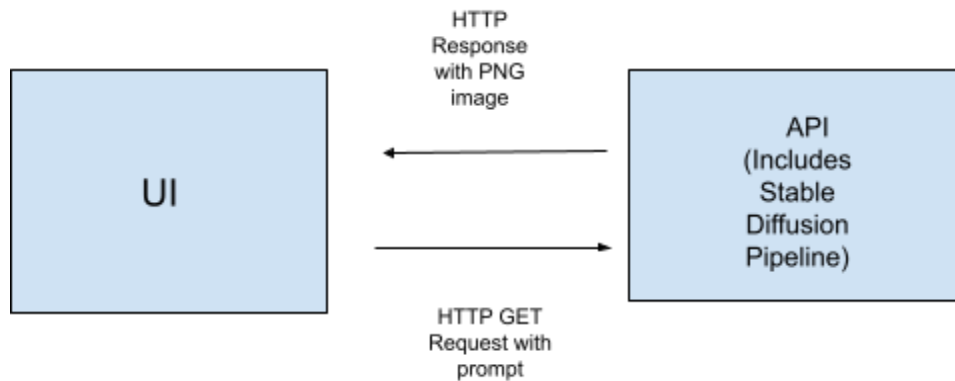
have budgeted for.

We ran a pretrained model (Stable Diffusion) sentence by sentence in the paragraph. Then, explored ways to further filter the result and outputs. We also took another approach to find and or generate a dataset with educational content for the subject of our choice to train the model.

At the end of the project we encountered errors with local Text Inversion installations and cloud resource constraints that prevented us from completing a custom checkpoint. This approach could be explored for future development with electronics or other subject areas.

Documentation on Implementation

UML Diagram of our two tier architecture for our ESL Stable Diffusion application.



The frontend was developed using vanilla Javascript, CSS, and HTML. The backend API is written in Python and implements the following libraries: flask, flask-cors, diffusers, and unicorn, transformers. The Stable Diffusion pipeline where we feed in our prompts is initialized in the API. We deployed our platform to Jarvis Labs AI, an affordable GPU cloud provider, any other cloud provider should be sufficient. In Jarvis Labs these were the specs for our instance.

GPU Type: RTX5000 x 1

RAM: 32GB, Cores: 7

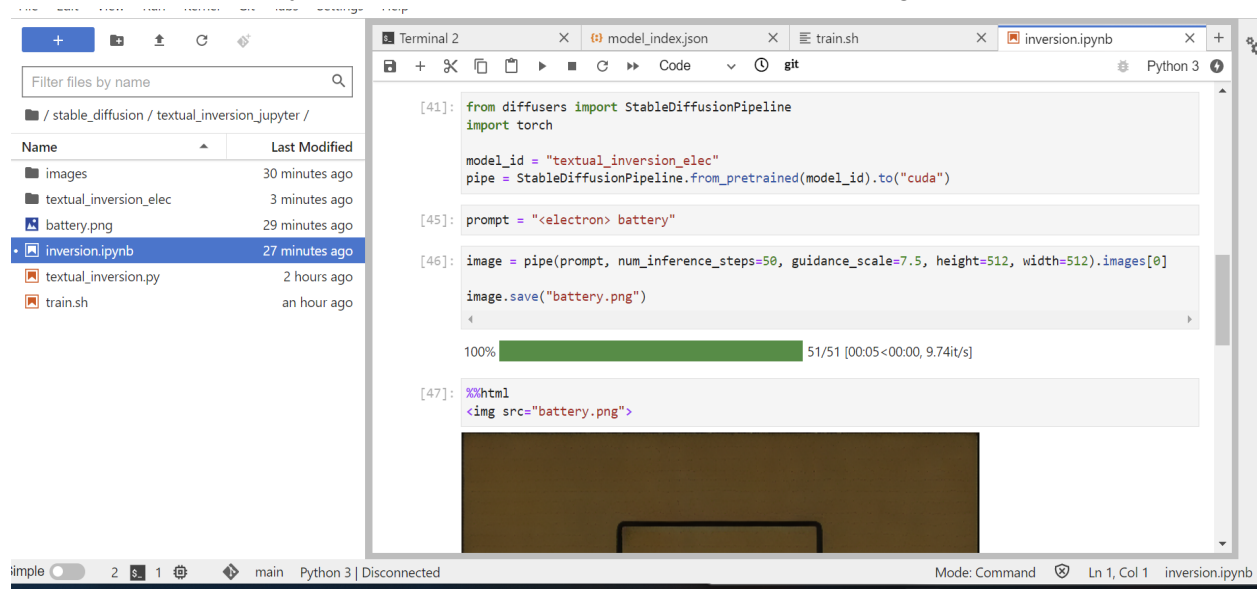
SSD 100GB

The code for our textual inversion implementation can be found in this folder.

https://github.com/mollyty/stable_diffusion/tree/main/textual_inversion_jupyter

Follow the instructions in the .ipynb file (jupyter notebook) to get started on development. We did not upload our custom checkpoint due to size requirements in Git and our checkpoint folder being between 3-4 Gbs. You will need to follow the instructions in the jupyter notebook to

generate your own checkpoint. Training took more than 1 hour and that was with our RTX5000 GPU. Running it is of course optional, in the folder we have a png file generated using our custom checkpoint, if you were to run it the flow would look something like this.



```
[41]: from diffusers import StableDiffusionPipeline
import torch

model_id = "textual_inversion_elec"
pipe = StableDiffusionPipeline.from_pretrained(model_id).to("cuda")

[42]: prompt = "<electron> battery"

[43]: image = pipe(prompt, num_inference_steps=50, guidance_scale=7.5, height=512, width=512).images[0]

[44]: image.save("battery.png")

[45]: 100% ██████████ 51/51 [00:05<00:00, 9.74it/s]

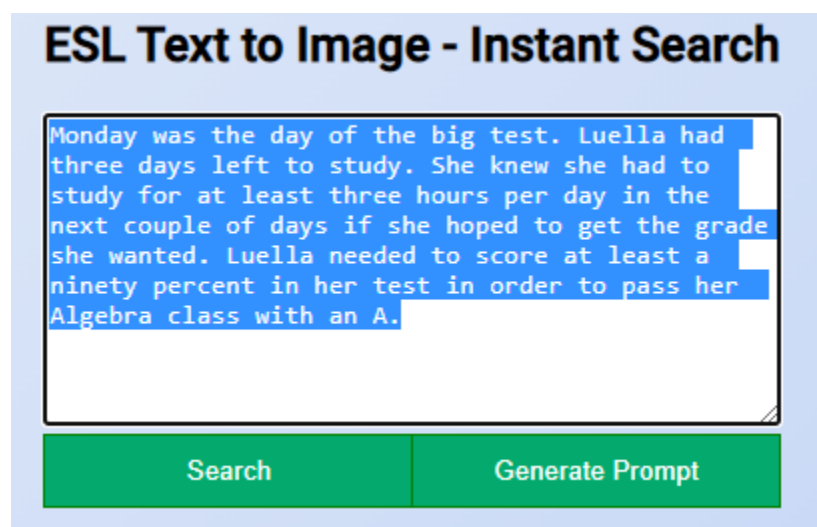
[46]: %html

```

If we had more time we would have connected our custom checkpoint to the pipeline initialized in our API. Doing that we would have started generating more specific and tailored images based on ESL or electronic images we used to train the model.

Documentation on the usage

Visual Topics Application:
cs410-sd-app.github.io



Enter a paragraph click "Generate Prompt" and click on search for our application to start generating images

For local development on the backend API or frontend, you can follow the instructions in this README file in our repo.

https://github.com/mollytyy/stable_diffusion/blob/main/app/README.md

If you solely wanted to test our API to return images you can hit the following endpoint. In the query parameter replace {insert_prompt_here} to your desired prompt. Don't forget to convert your string to a URL Search parameter.

https://notebooksa.jarvislabs.ai/UbU2hKCvajiAcoK_S70CPzmBgYGa-KY_xQKWmujmaZe3p5b9kpDYEbCE0vhgNdeU/image?q={insert_prompt_here}

The instructions for our textual inversion implementation can be found in this folder.

https://github.com/mollytyy/stable_diffusion/tree/main/textual_inversion_jupyter

Stable Diffusion usage:

https://github.com/mollytyy/stable_diffusion#readme - see "Reference Sampling Script".

Note: Cloud implementation has one GPU provisioned for the application backend, therefore when submitting image generation sequentially to avoid resource contention

Jarvis Labs usage:

If you want to deploy your own Flask API or FastApi to Jarvis labs you can follow this guide.

<https://jarvislabs.ai/docs/deploy/flask/>

The rest will just be initializing your stable diffusion pipeline within the app and creating an endpoint for it.

<https://jarvislabs.ai/docs/deploy/flask/>

Software Install

(Optional for grading, the deployed site demonstrated stable diffusion functionality) Complete installation instructions to run locally and usage documentation can be found in the readme here:

https://github.com/mollytyy/stable_diffusion#readme

Team Member Contribution

Molly Yang

Researched stable diffusion and its applications. Gained understanding on where and how it has been used and the ethics behind text to image technologies. Attempted creating a custom checkpoint locally, but needed much more computational resources. Surveyed toolkits that improve prompt generation. Collected and modified training images for electronics components to be used in textual inversion.

Louie Hernandez

Spent time researching different low memory stable diffusion forks to use for my CPU-only machine. Contribution included developing the frontend application for the project as well as developing the backend api for the platform as well. Integrated stable diffusion pipeline into our api. Deployed both frontend and backend to a cloud provider. Developed and implemented the textual inversion custom checkpoint and created a jupyter notebook for it as well.

Jason Lundquist

Contribution included Team Captain, coordinated team meetings, project status updates to CMT for assignments. Conducted research and proof of concept testing of Stable Diffusion and Text Inversion models on local hardware. Established baseline with Stable Diffusion model to use as comparison against the goal to use Text Inversion to customize the model in line with educational scope and constraints.