CSCE 5290: Natural Language Processing

Project Increment 1

GitHub link: https://github.com/RyanTolbert/ArtGenerator

Project Description:

This project will create an album cover art from a user-entered prompt. The user can customize this cover using basic image adjustments, like color saturation, tint, a text editor, etc.

1. Project Title and Team Members
   Title: Album Cover Generator
   Team: Ryan Tolbert

2. Goals and Objectives:
   - **Motivation:** I am a graphic designer, photographer, and music enthusiast and would like to see if I can use NLP to create an album cover from a prompt. I think this tool could be used to inspire or generate a piece of concept art.

   - **Significance:** This album cover generator can be used as a tool for inspiration and idea creation. Having AI create an image can provide a new view of the world. These images may not look particularly realistic, but they can provide the necessary proof-of-concepts for certain ideas. Using this people can brainstorm ideas for album covers and customize the cover art to their liking. There has been text to image generators before, but there are no art generators. The main idea for this is to create cover arts for music albums, but you could use it for any sort of art creation. I aim to create a useful tool for artists and designers to use to create art.

   - **Objectives:** The user will enter a prompt of what they want the cover art to look like. A prompt can be something like "A dog wearing a red hat". Using NLP, my program will determine what the user wants and use the OpenAI model to generate an image based on the prompt. The user will also be given options on what they would like to add or change with the cover art. So, if the user wants the image to be black and white, they can do that. Perhaps the user wants to add the album title in text on the cover art, they should be able to do that. This tool should provide some basic tools to generate and customize a cover art that they like.

- **Features:** The main feature is that you can input any prompt and get an image output. There should be a feature to add text and select the font, size of the text, and position of the text on the cover art. There should also be an option to make the cover art black and white or add saturation to make the colors brighter. I plan to also add more basic image options to allow for the most customization to the user. I also may provide an option for the user to use their voice to enter a prompt. This would enable users to enter a prompt verbally or via text.

Increment 1:

- **Related Work (Background):** There currently exists methods to make an album art cover, but it requires you to upload your own photo. Alternatively, there also exists ways you can generate an image from scratch with a user inputted text prompt. However, you can not edit these generated images. I am combining both of these ideas to make for an art generator that allows the user to type what they want the photo to be of and also some basic photo editing options to adjust the image to the user's liking.

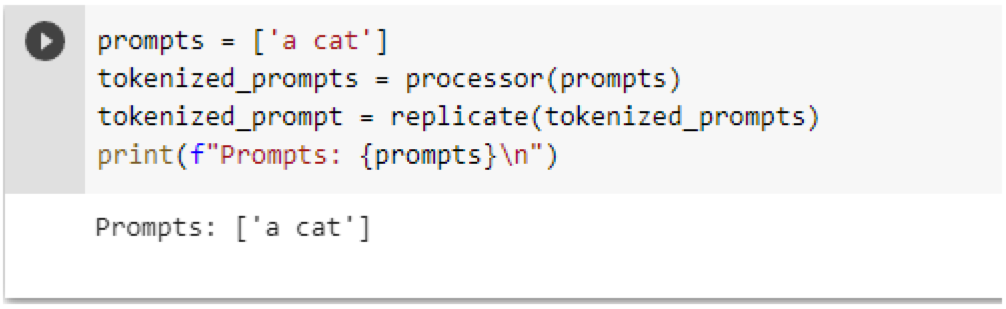
- **Dataset:** I am using the Hugging Face dataset to process the text that will be inputted from the user. With that, I am using the DALL-E model to build an image from the processed text. I am also using Vqgan-JAX to decode the images. All of these are listed and linked in the references of this document.

- **Detail Design of Features:** For this increment the feature added is the generation of an image from a user-inputted prompt.

- **Analysis:** Currently, the program is output accurate images. An interface for the user to input this prompt would be ideal. In its current state, the user must type their prompt in the command line. A level of abstraction could make this program much better for the user.

- **Implementation:** For the implementation, I did tests with hugging face first. The reason for this is to ensure that the text was being tokenized and analyzed properly. From there I separately test the generation of images with premade text prompts. Once that was working well, I combined these functions. This makes for a program that takes an prompt from the user and builds an image based on their input.

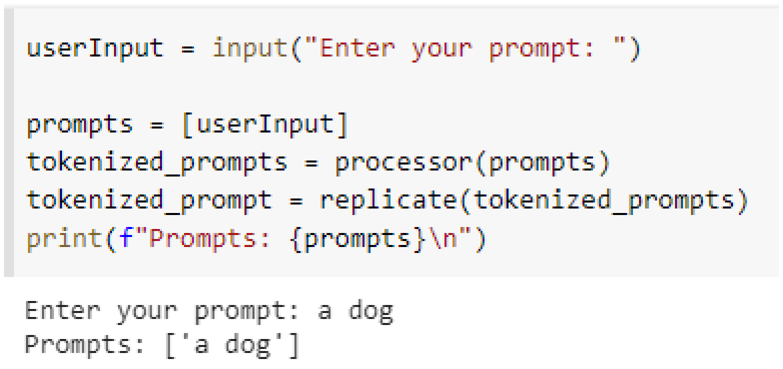
- **Preliminary Results:** First testing the processing of text:



```
prompts = ['a cat']
tokenized_prompts = processor(prompts)
tokenized_prompt = replicate(tokenized_prompts)
print(f"Prompts: {prompts}\n")
```

```
Prompts: ['a cat']
```

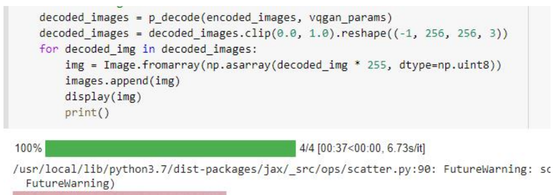
Testing a user inputted text:



```
userInput = input("Enter your prompt: ")

prompts = [userInput]
tokenized_prompts = processor(prompts)
tokenized_prompt = replicate(tokenized_prompts)
print(f"Prompts: {prompts}\n")
```

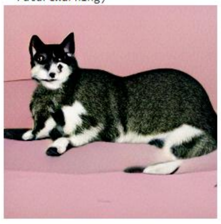
```
Enter your prompt: a dog
Prompts: ['a dog']
```

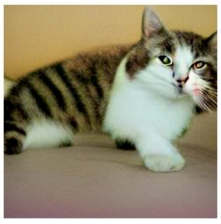
The first tests with a premade input:

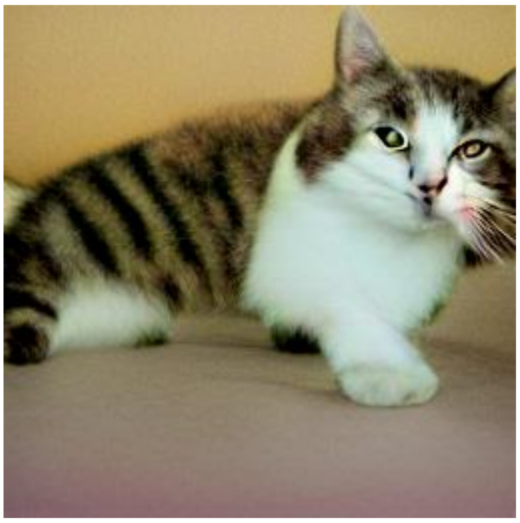


```
    decoded_images = p_decode(encoded_images, vqgan_params)
    decoded_images = decoded_images.clip(0.0, 1.0).reshape((-1, 256, 256, 3))
    for decoded_img in decoded_images:
        img = Image.fromarray(np.asarray(decoded_img * 255, dtype=np.uint8))
        images.append(img)
        display(img)
        print()
```

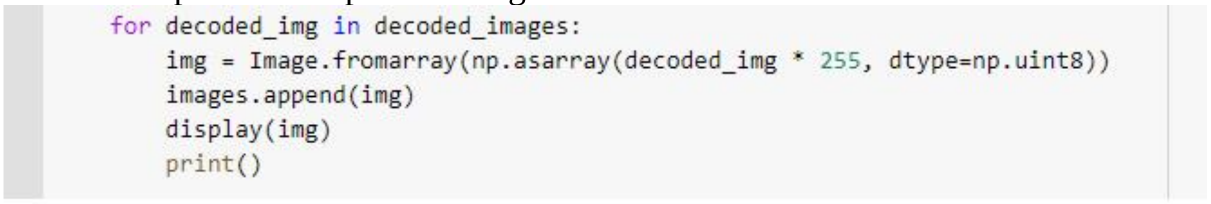
```
100% 4/4 [00:37<00:00, 6.73s/it]
/usr/local/lib/python3.7/dist-packages/jax/_src/ops/scatter.py:90: FutureWarning: sc
  FutureWarning)
```





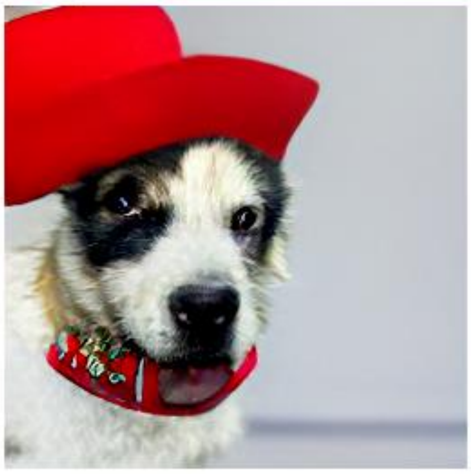


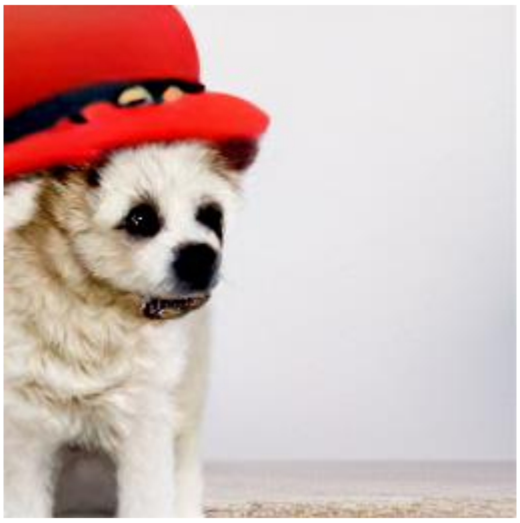
This result is pretty good, it is very clear that the image is of a cat. Now to test with a more complex user-inputted string:



```
    for decoded_img in decoded_images:
        img = Image.fromarray(np.asarray(decoded_img * 255, dtype=np.uint8))
        images.append(img)
        display(img)
        print()
```

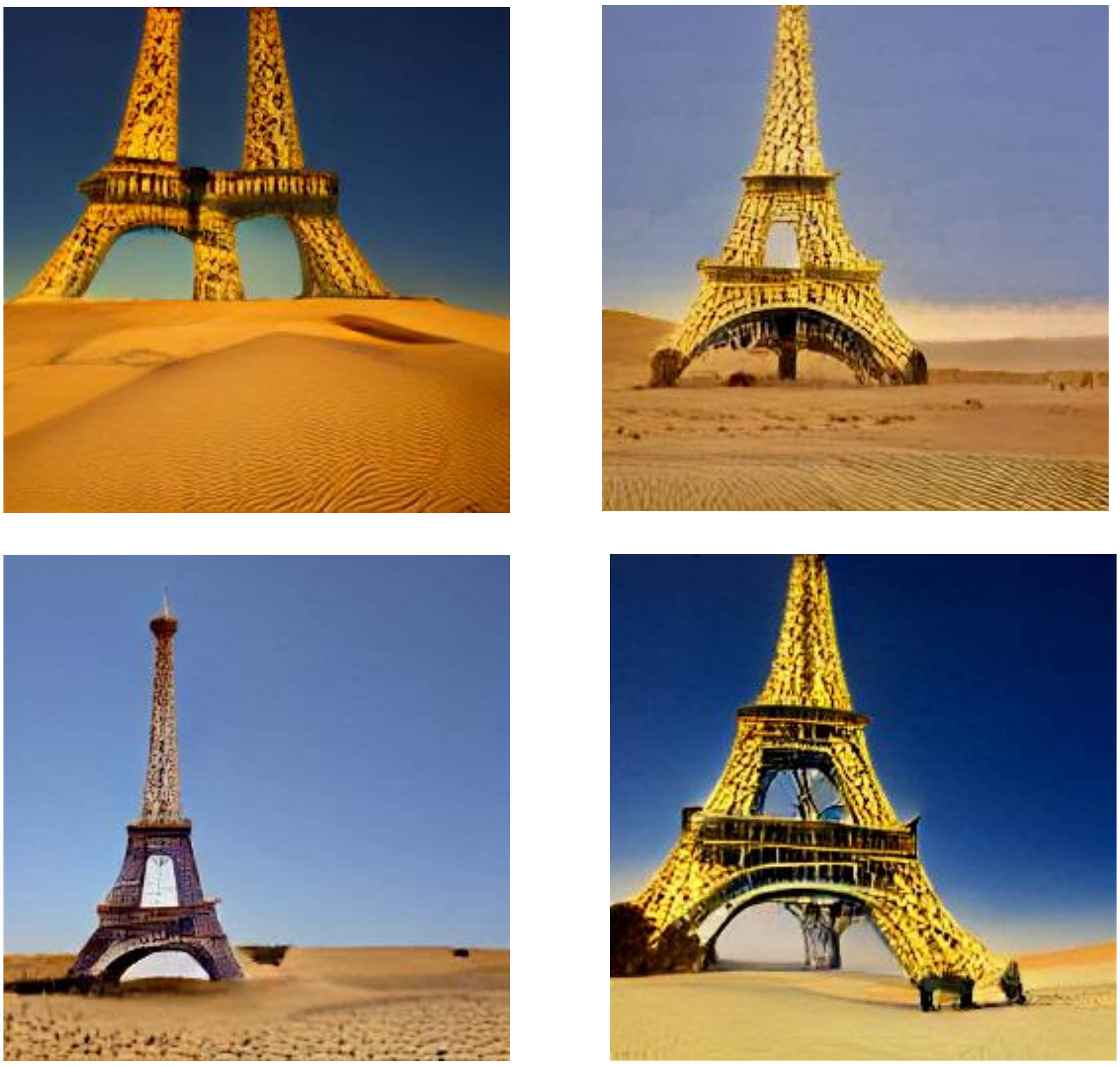
```
Enter your prompt: a dog with a red hat on
Prompts: ['a dog with a red hat on']

100% ████████████████████ 4/4 [00:12<00:00, 3.18s/it]
```





I was impressed with these results. The prompt inputted was “a dog with a red hat on” and the results are quite clear given the complex input. It is easy to tell it is a dog and the hat isn’t the best, but it is red. So, for the most part, these are satisfactory results.

Finally, I tested this with a very complex prompt to see its current capabilities. I set it to output 4 different photos of the same prompt. The prompt entered was “The Eiffel Tower in the Sahara Desert” and here were the results:



This is an excellent result, especially given the tough input prompt. At this point I was satisfied with all the results and started planning towards implementing the image options.

- **Project Management:**
    - **Implementation status report**
        - **Work completed:** Generate an image from user-inputted text.
            - *Description:* The user enters a prompt and the program generates and image from scratch based on the prompt. For example, if the user enters; “a cactus”, the program will build an image of a cactus.
            - *Responsibility:* Ryan Tolbert
            - *Contributions:* Ryan Tolbert
        - **Work to be completed:** Allow the user to make adjustments to the image.
            - *Description:* With the image generated from the program, the user should be able to make basic image adjustments, like saturation, hue shift, adding text, adding a border, etc. For example, if the user wants the image to be in black and white with a white border, they should have the options to do so.
            - *Responsibility:* Ryan Tolbert
            - *Contributions:* Ryan Tolbert

3. References

This is the model I am using to generate images from the text: https://github.com/openai/DALL-E

I am using this decoder: https://github.com/patil-suraj/vqgan-jax

I am using hugging face to process the text: https://huggingface.co/blog/how-to-generate

GitHub link: https://github.com/RyanTolbert/ArtGenerator