CS6365 Final Project Report

LIKE: A Comparative Study of AI Image Generators Through CLIP Semantics and Multi-Faceted Output Analysis

Rohit George

July 24, 2025

Contents

1. Abstract

This project aims to explore how different image generation models interpret and encode natural language prompts in visual outputs. To do this, prompts will be provided to these models and images will be generated and stored for each. These images will be analyzed and compared to evaluate how stylistic and semantic features are represented in high-dimensional latent spaces. Clusters and patterns will be studied to understand the differences in prompt interpretation and stylistic choices across models.

2. Introduction and Motivation:

In the past few years, there have been major strides in image generation models, with many becoming open-source, allowing for users across the internet to freely utilize them [1]. Specifically, OpenAI's DALL-E, Adobe's Firefly, and Canva are new AI models that tackle this challenge using their own styles [1]. Furthermore, models such as Stable Diffusion are released on huggingface, allowing for easy access [2]. All of these possess the ability to convert text based queries into images, allowing users to express their ideas and creativity in visual form.

However, there is still limited understanding of how these diverse models internally represent and differentiate between semantic content and style. For example, while two models may produce images that appear similar at first glance, a deeper look could reveal significant differences in object identification, texture, and style [1]. As a result, people may not use the model that best serves their needs and vision, preventing them from making what they desire.

There are also ethical considerations, as AI art has been seen as a major threat to creativity, IP ownership, and job security [3]. This is because it is trained on previous, human made, artwork, thus potentially violating IP constraints in its production. Its ability to quickly create outputs for free has lessened the need for outsourcing and commissioning artwork to actual artists. By training on past artwork and lessening the requests for new art, AI art has also been seen as a threat to creativity overall. As such, many dislike the concept of AI art and would like to know when they are looking at art that is AI generated or human generated. This study could aid in detecting AI art, understanding model biases and reporting when they are detected.

Finally, better understanding the outputs of these models can help detect where improvements need to be made. Research into these models has shown that understanding the visual output of a modal helps reveal its biases and weaknesses [4]. Specifically, a study of a model's semantic fidelity (how accurately it translates textual nuances into visual details), stylistic consistency (whether it consistently encodes stylistic choices across varied prompts), and model differentiation (do different models have different strengths or limitations) can help advance our knowledge of where these

models succeed and fail, thus informing on what needs to be targeted to improve the model.

Overall, this project aims to provide a quantitative study to assess and compare latent representations. By doing so, it hopes to inform users of AI art, allowing them to best know which model to use for their own image generation and detect when they are looking at art generated by an AI or a human.

Leonardo AI [1]

Canva [1]

3. Related Work

With the issues and concerns facing image generation models, a growing body of research has emerged to better understand their internal mechanisms, limitations, and outputs. Recent studies have started to focus on interpreting latent representations and identifying potential biases, specifically focusing on how text-to-image models encode semantic information and how biases may emerge in image generation and text evaluation. To do this, researchers have started to use multimodal embedding models such as CLIP to analyze how generative models encode data.

1. *Bias Begets Bias: The Impact of Biased Embeddings on Diffusion Models [4]*
   This study looks into how biased text embeddings can propagate into image generation models. More importantly, it highlights how biased evaluation metrics, such as CLIP, can reinforce these same biases, lowering scores for balanced models and rewarding “unfair behavior” [4]. As such, they highlight bias mitigation methods and the need for interpretability tools that extend into latent space visualizations.

2. *Discovering and Mitigating Biases in CLIP-based Image Editing [5]*
   This work demonstrates how CLIP embeddings can amplify social biases when used for editing or altering images, discussing methods on how to detect and mitigate these effects. While the focus is on image editing, the main idea of how CLIP’s latent space reflects societal biases is very relevant to how I interpret the outputs of text-to-image models.

3. *Fairness and Bias in Multimodal AI: A Survey [6]*
   This survey outlines the challenges in ensuring fairness in multimodal systems, especially those combining vision and language. It highlights the lack of systemic tools for visualizing and interpreting the internal representations of generative models. This project will address this need by applying dimensionality reduction and clustering techniques to analyze the structure of generated images, thus making the systems more transparent and interpretable.
4. *A Comparative Study of GAN-Generated Handwriting Images and MNIST Images using t-SNE Visualization [7]*
   While this focuses on GAN-generated images, which is not the focus of this project, it evaluates them using t-distributed stochastic neighbor embedding (t-SNE) visualization. Overall, it was found that the generated images had some differences in the distribution of the features when compared to the original images. This method can be used to compare the generated images and their object distribution.

5. *Assessing Sample Quality via the Latent Space of Generative Models [8]*
   This paper studies the latent space of a model, introducing a "latent density score". This demonstrates how latent structure can help quantify sample quality and will allow me to compare the sample quality of each model used.

4. System Architecture

The system architecture consists of a pipeline designed to evaluate and compare these models and can be classified into these key stages:

1. Prompt Generation:
   - A curated set of prompts is created, varying across objects, styles, and contexts. These prompts serve as consistent inputs across all generation models.
2. Image Generation:
- For each prompt, images are generated using multiple generative models.
   - Stable Diffusion via HuggingFace
   - DALL-E via OpenAI
   - Leonardo AI
   - Canva
3. Embedding:
- Generated images are passed through CLIP to extract 512-dimensional image embeddings that encode both semantic and stylistic information.
4. Dimensionality Reduction and Clustering:
- Embeddings are reduced to 2D using t-SNE to visualize structure. Clustering techniques can also be applied to identify natural groupings in latent space.

5. Evaluation and Visualization:
- A set of plots and analyses are generated. Scatterplots will be made of embeddings. Image grids grouped by cluster or prompt. Bar charts comparing prompt distances and CLIP similarity scores between prompt text and image will also be made.
6. Report:
- All findings, visuals, and metrics will be compiled into the final report discussing prompt fidelity, stylistic consistency, model differences, and potential biases in latent representations.

The hardware required to do these includes:

1. A modern laptop or desktop with a GPU:
- Having a GPU with a proper device will greatly accelerate model generation and comparison times.
2. Cloud computation resources on Google Colab:
- Connecting to a Nvidia GPU on Google Colab will also be extremely useful in lessening runtimes.
3. Sufficient Storage:
- A sufficient amount of memory storage will be required to store generated images, embeddings, and model weights.

The software required:

1. Python 3.8+:
- Required to run imported libraries:
2. HuggingFace:
- Can be used to import and access pretrained models such as Stable Diffusion and CLIP.
3. API Access:
- Necessary to access certain models, such as DALL-E, which will require OpenAI API.
4. CLIP:
- Necessary for embedding image outputs into latent space. Can be accessed through HuggingFace.
5. Dimensionality Reduction Libraries:
- Required to use t-SNE to compare models.
6. Visualization Tools:
- Matplotlib and Seaborn will be needed for comparing and visualizing results.
7. Google Colab:
- Will be utilized for running and presenting the full pipeline.

5. Dataset

5.1 Data Motivation and Purpose:

In the past few years, generative image models have rapidly advanced in both quality and accessibility. Models like OpenAI's DALL-E, Midjourney, and Stable Diffusion have become widely adopted for both personal and business usage across many fields. However, despite their popularity, users rarely know how these models interpret and respond to textual prompts. For any given prompt, these models may produce greatly different outputs, varying in terms of style, content, composition, placement, and fidelity.

This raises a core question: how consistently do different generative models interpret the same prompt? While user preferences and model reviews often center around realism, style, or visual appearance, few resources exist that prove a systemic way to compare how models semantically respond to the same inputs. This project seeks to address this gap by constructing a dataset designed for prompt-level comparison. By generating images from five major generative models using the same set of 20 prompts, each repeated across three trials, the dataset seeks to isolate the prompt as a controlled variable, allowing a better analysis on how model behavior diverges or converges in response.

Unlike traditional image generation benchmarks that emphasize realism or human ratings, this dataset is analyzed using CLIP embeddings, which provide a high-dimensional semantic representation of each image. This allows for a range of quantitative comparisons across models including cosine similarity, latent space clustering, and prompt-level variation. As a result, this dataset offers a new lens for evaluating generative models, focusing on their underlying interpretation and semantic fidelity to user intent. This dataset thus aims to serve as a foundation for deeper insights into model behavior, robustness, and alignment. It also has broader implications for users seeking to understand which models best suit their goals and may aid them in identifying model generated images through common patterns. Such a feat would help differentiate artificially generated images from real images in a time of confusion regarding deepfakes and fake news generated through AI.

5.2 Dataset Construction and Design:

The dataset was constructed to meet three design goals:

1. Control for prompt variation.
2. Sample outputs across trials.
3. Capture diversity across multiple different models.

Each of these goals informed decisions on how prompts were selected, how images were named and stored, and how models were integrated into the pipeline.

The dataset consists of 20 distinct prompts, each crafted to be quite general and open-ended, ranging from fantasy and whimsical (“A teddy bear having tea with a dragon”) to more realistic (“a snowman standing on a beach in the middle of summer”). At the same time, they sought to avoid being completely realistic, encouraging model creativity. As such, to avoid choosing random or repetitive phrases, prompts were categorized into five thematic groups, each targeting a different area of model generation:

1. Animals in Usual Contexts: Tests how models reimagine animals.
2. Urban + Nature: Blends architectural and natural elements in imaginative ways.
3. Style Comparison: Tailored for observing artistic style, abstraction, and rendering techniques.
4. Human + Sci-fi: Focuses on human subjects in futuristic settings.
5. Whimsical/Fantasy: Highly imaginative prompts designed to stretch the limits of creativity.

This prompt categorization serves a few purposes:

1. Enables controlled comparisons across models for each type of visual and conceptual challenge.
2. Creates opportunities for cluster-based and style-based analyses later in the pipeline.
3. Balances between semantic clarity for CLIP embedding evaluation and visual diversity, ensuring the dataset doesn’t overfit to a narrow visual domain.

For each prompt, three separate generations were created per model to capture output variance, resulting in 15 images per prompt (5 models across three trials). As such, the final dataset contained 300 images, each saved in a consistent .png format and organized into model-specific folders.

To conduct a rigorous comparison between state-of-the-art generative image models, the dataset was constructed to meet several core design goals: (1) control for prompt variation, (2) sample outputs across trials, and (3) capture diversity across multiple model families. Each of these goals informed decisions on how prompts were selected, how images were named and stored, and how models were integrated into the pipeline.

The final prompt list was as such:

Prompt ID	Prompt Text	Category
0	A cat wearing medieval	Animals In Unusual

	armor standing in a battlefield	Contexts
1	A golden retriever playing chess at a table	Animals In Unusual Contexts
2	A frog wearing a spacesuit on the moon	Animals In Unusual Contexts
3	An owl reading a book under a streetlight	Animals In Unusual Contexts
4	A futuristic city skyline with flying cars at sunset	Urban + Nature
5	A treehouse built on top of a skyscraper	Urban + Nature
6	A mountain range made of glass and crystal	Urban + Nature
7	A forest with glowing neon mushrooms	Urban + Nature
8	A robot painting a self-portrain in Van Gogh style	Style Comparison
9	A street market scene in watercolor	Style Comparison
10	A steampunk-style train crossing a desert	Style Comparison
11	A jazz band in an abstract Picasso-style painting	Style Comparison
12	A woman exploring an alien jungle on a distant planet	Human + Sci-Fi
13	A person floating through a digital cyberspace	Human + Sci-Fi

	landscape	
14	An astronaut relaxing on a hammock between two planets	Human + Sci-Fi
15	A medieval knight using a laptop in a castle library	Human + Sci-Fi
16	A teddy bear having tea with a dragon	Whimsical/Fantasy
17	A carousel made of candy and gingerbread	Whimsical/Fantasy
18	A sushi roll flying through outer space	Whimsical/Fantasy
19	A snowman standing on a beach in the middle of summer	Whimsical/Fantasy

Table[1]: Table of prompt IDs, prompt text, and prompt category.

5.3 Model Selection:

To ensure a comprehensive and representative analysis, five generative models were chosen for inclusion in the dataset based on their popularity, accessibility, and design philosophies. The models included were:

Model	Developer	Access Type
DALL-E 3	OpenAI	Closed(Web UI), free using chatgpt.
Adobe Firefly	Adobe	Closed(Web UI), used $10 a month tier.
Midjourney V6	Midjourney	Closed(Web UI), used one month free trial.
Stable Diffusion	Stability AI	Open(Hugging Face)

Kandinsky 2.1	Sberbank	Open (Hugging Face)

Table[2]: Table of the models used, their creators/developers, and how they are accessed.

The models were selected due to their wide popularity, as they are some of the most popular image generation models [9] and easily accessible. Furthermore, they are known for having stylistic differences, which this project aimed to properly test. For example, DALL-E is known for overall realistic quality while midjourney is known for more artistic results. The prompt list was thus affected to also highlight these differences, with the expectation that DALL-E and Midjourney would differ greatly in the more realistic (snowman in summer) prompts and more fantasy-focused prompts (teddybear having tea with a dragon).

This diversity thus makes the dataset useful for understanding how stylistic biases and model design choices will affect the semantic interpretation of identical prompts.

5.4 Naming and Trial Structure:

Each of the 20 prompts in the dataset was rendered three times per model, yielding 15 images per prompt and a total of 300 images across the dataset. This repeated trail design was aimed to capture intra-model variance (how much a model's outputs vary for the same prompt) and enable prompt-level and trail-level comparison between models.

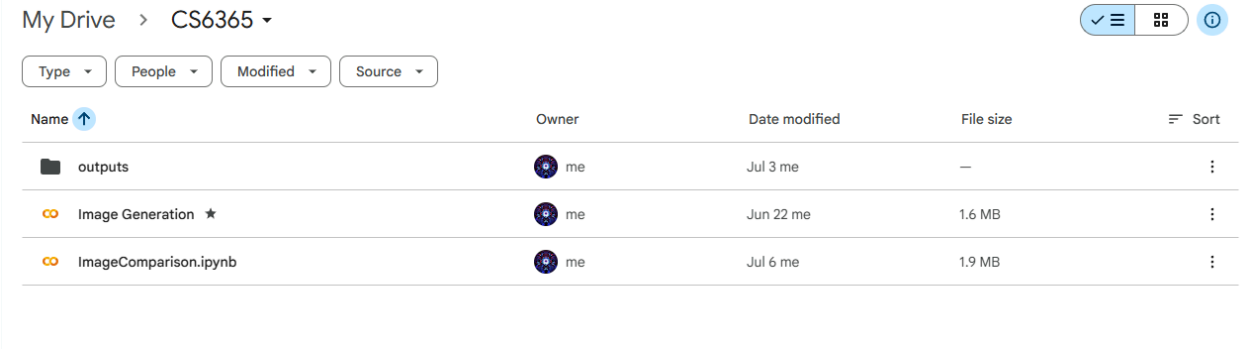
All image filenames follow a standardized naming convention:

- Format: {model_name}{prompt_id}.{trail_number}.png.
    - Each image filename was labeled to reflect the prompt ID and trial number, enabling for easy indexing and analysis. For example, DALL-E_0.0.png refers to model DALL-E prompt ID 0 (cat in armor), and trial 0.
    - This structure allows for both model-level and prompt-level aggregation and comparison, which later becomes essential for computing and comparing CLIP embeddings, clustering, and prom-twise heatmaps. Overall, it allows for:
        - The quick slicing by model, prompt ID, or trials using code.
        - Aggregated statistics and comparisons by prompt level.
        - Alignment with CLIP embeddings and dimensionality reduction workflows.
    - Care was taken to ensure consistent image dimensions across models to allow valid embedding comparisons. While exact aspect ratios may differ slightly due to model defaults.

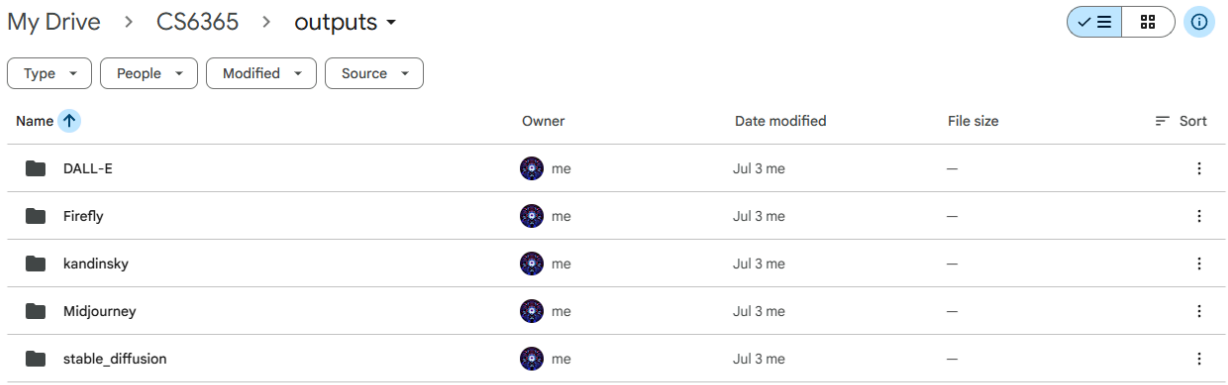
By enforcing this structure across models and generation, the dataset is robustly designed for extensibility, automation, and reproducibility in future experiments.

5.5 File organization, Preprocessing, Prompt Generation:

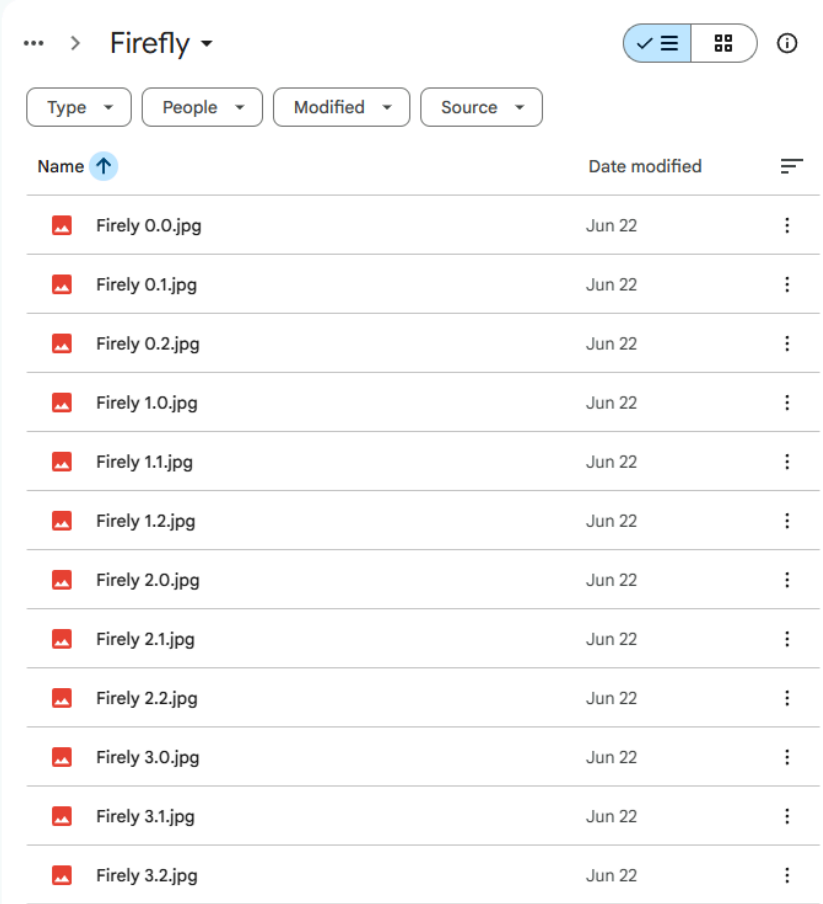
All work was done and stored in google drive, under MyDrive/CS6365. Doing this allowed me to work in Google Colab, thus allowing me to make use of hosted runtimes to increase the running speed of the code.



To ensure maintainability and consistency throughout the project, all images were stored using a clear and hierarchical folder structure within a Google Drive directory CS6365/Outputs. Inside this folder, each model had their own respective folder to store the 60 images generated through using them.



For example, MyDrive > CS6365 > outputs > Firefly looks as such:



To populate these files, each model was used to generate 20 prompts across three trials.

5.5.1 Kandinsky 2.1:

To do this for Kandinsky 2.1 and stable_diffusion, the file MyDrive/CS6365/Image Generation.ipynb was used. The image generation process involved two stages:

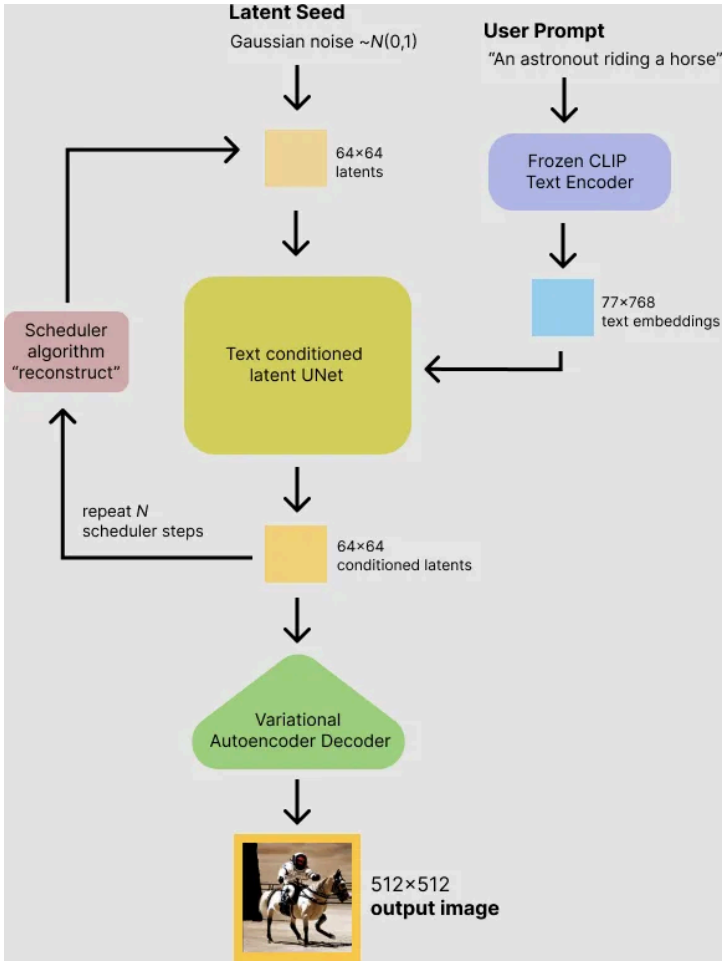
1. Prior Pipeline (KandinskyPriorPipeline) - Converts the text prompt into image embeddings, essentially translating the text prompt into a latent image embedding that represents what the model thinks the image should look like. This contains both positive embeddings (what should be emphasized in the image) and negative embeddings (what should be avoided in the image). This dual-guidance setup gives more control over image outputs.
2. Decoder Pipeline (KandinskyPipeline) - Uses the embeddings generated by the prior pipeline to generate an image that highlights the positive embeddings and avoids the negative embeddings. A guidance scale parameter is set that controls the influence of the prompt during generation. I set mine at 4.0, which is relatively medium as I wanted a balanced output.

Each prompt was looped over three times to account for variability and the output was saved using a standardized naming convention (kandinsky_2_1_<prompt_id>_<trial>.png.
The code used for generation was:

```
from diffusers import KandinskyPipeline, KandinskyPriorPipeline
import torch, os
from PIL import Image

prior = KandinskyPriorPipeline.from_pretrained(
    "kandinsky-community/kandinsky-2-1-prior", torch_dtype=torch.float16
).to("cuda")

decoder = KandinskyPipeline.from_pretrained(
    "kandinsky-community/kandinsky-2-1", torch_dtype=torch.float16
).to("cuda")
for idx, prompt in enumerate(prompts):
    for i in range(3):
        #Get both positive and negative embeddings
        prior_output = prior(prompt)
        image_embeds = prior_output.image_embeds
        negative_embeds = prior_output.negative_image_embeds

        #Decode the image using both embeddings
        result = decoder(
            prompt=prompt,
            image_embeds=image_embeds,
            negative_image_embeds=negative_embeds,
            guidance_scale=4.0
        )
        image = result.images[0]

        #Save to output folder
        filename = f"kandinsky_2_1_{idx}_{i}.png"
        image.save(os.path.join(kandinsky_dir, filename))
```

5.5.2 Stable Diffusion:

Image Generation.ipynb was similarly used for Stable Diffusion image generation. For Stable Diffusion, a simpler StableDiffusionPipeline was used from the Hugging Face diffusers library. Unlike Kandinsky's two-stage approach, Stable Diffusion was a single unified model that directly converts a text prompt into an image through a denoising diffusion process. Specifically, it compresses an image into latent space and then runs a reverse diffusion process to remove noise [10]. The main idea is that it begins with pure noise in the latent space and removes this noise step by step through a set of denoising steps (reverse diffusion), which are guided by a U-Net neural network conditioned on the text prompt [11]. Conditioning is achieved by using CLIP, which encodes the text to create embeddings that the model can understand. After the reverse diffusion process is finished, the final latent representation is passed through a variational autoencoder (VAE) that converts it into a pixel space image.



Figure[1]: How Stable Diffusion Works [11]

Each of the 20 prompts was passed through the model three times through the usage of a loop and the resulting image was saved with the format sd_<prompt_id>_<trial>.png.

```
!pip install diffusers transformers accelerate --upgrade

from diffusers import StableDiffusionPipeline
import torch, os

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

prompts = [
    "A cat wearing medieval armor standing in a battlefield",
    "A golden retriever playing chess at a table",
    "A frog wearing a spacesuit on the moon",
    "An owl reading a book under a streetlight",
    "A futuristic city skyline with flying cars at sunset",
    "A treehouse built on top of a skyscraper",
    "A mountain range made of glass and crystal",
    "A forest with glowing neon mushrooms",
    "A robot painting a self-portrait in Van Gogh style",
    "A street market scene in watercolor",
    "A steampunk-style train crossing a desert",
    "A jazz band in an abstract Picasso-style painting",
    "A woman exploring an alien jungle on a distant planet",
    "A person floating through a digital cyberspace landscape",
    "An astronaut relaxing on a hammock between two planets",
    "A medieval knight using a laptop in a castle library",
    "A teddy bear having tea with a dragon",
    "A carousel made of candy and gingerbread",
    "A sushi roll flying through outer space",
    "A snowman standing on a beach in the middle of summer"
]

for idx, prompt in enumerate(prompts):
    for i in range(3):
        image = pipe(prompt).images[0]
        filename = f"sd_{idx}_{i}.png"
        image.save(os.path.join(sd_dir, filename))
```

Figure[2]: Shows code where Stable Diffusion was used to generate prompt images. Contains the pipeline that uses a pretrained Stable Diffusion Pipeline, prompt list, and loop that iterates through the prompt list to create three images per prompt.

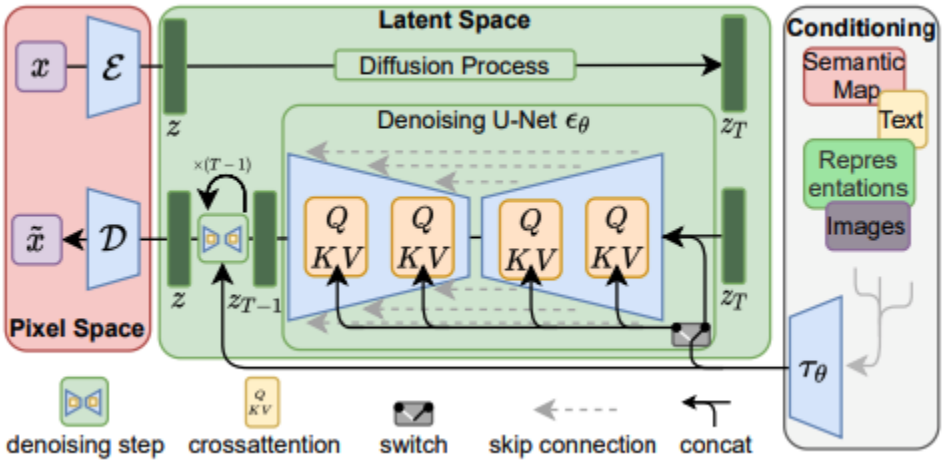
5.5.3 DALL-E 3:

DALL-E 3's image generation is built by combining CLIP, U-Net, and a VAE decoder [12]. When a user submits a prompt, the model first leverages CLIP to deeply understand and encode the semantic content of the text. This transforms the prompt into a text embedding that captures its literal and contextual meaning, allowing the model to know what the image should be like. Next, the U-Net starts with pure noise and, using guidance from CLIP-generated text embeddings, begins a diffusion-based denoising process. It basically creates a low-resolution version of the image by iteratively refining this noisy input to align with the prompt's content. This content is then passed through a VAE decoder, which scales it up to a high-resolution, detailed visual output.

To create these images, I went to OpenAI's ChatGPT. Specifically, I went to the Library tab, where past images can be viewed and new images can be created, located

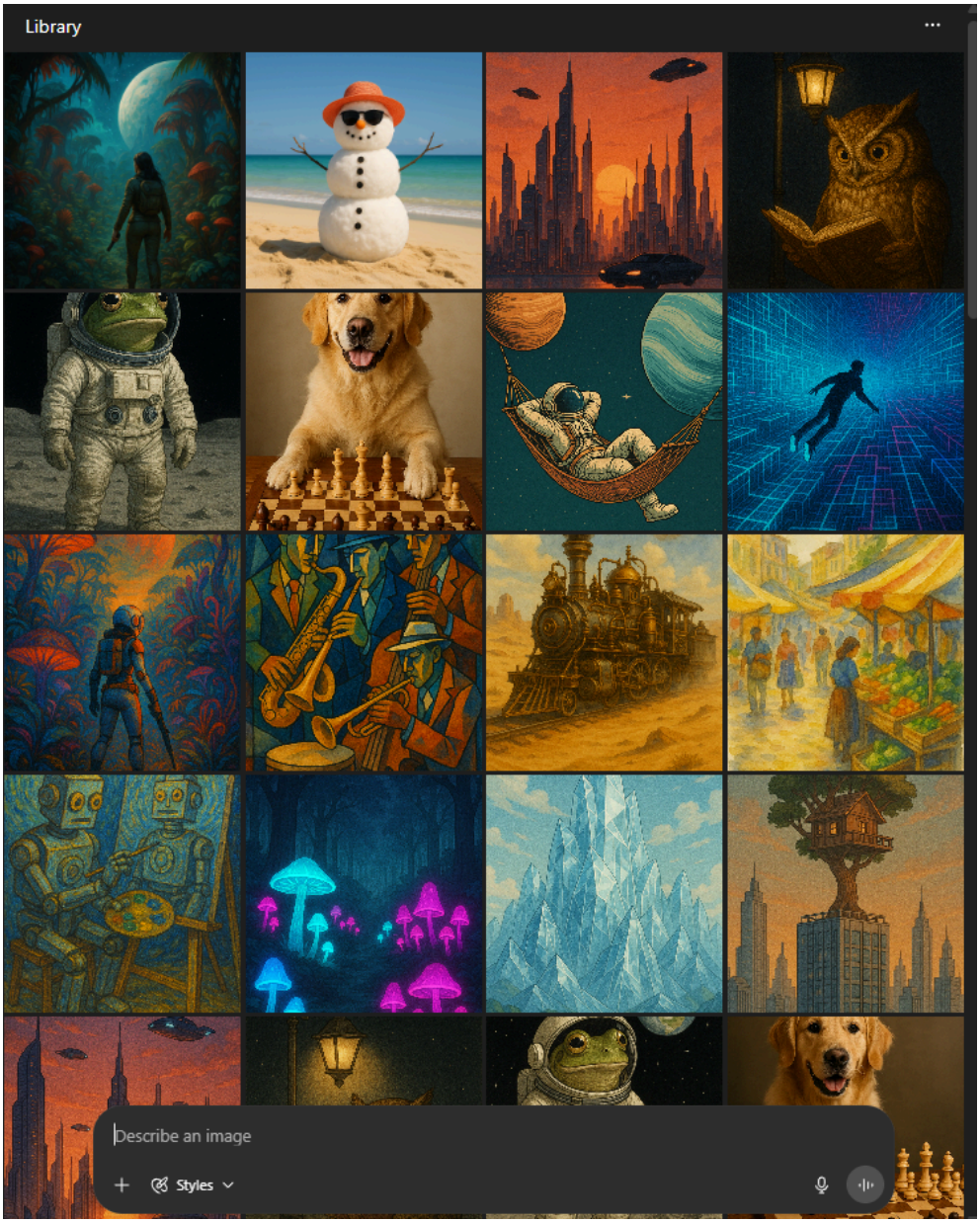
at https://chatgpt.com/library. In this page, in the “Describe an Image” text box, I entered each of the 20 prompts one by one, three times in total.

I then saved the produced images locally, which were then uploaded to Github and my Google Drive folder once all 60 images were produced.



Figure[3]: DALL-E unet architecture used to build and train a diffusion model. The model takes a clear pixel space image and adds noise to it. The model trains by attempting to remove the noise and recreate the original image.

[13]



Figure[4]: DALL-E Web Interface For Image Generation

5.5.4 Midjourney v6:

Midjourney is a generative AI system that creates images from natural language prompts, operating through a diffusion-based architecture [14]. It follows a similar process to other text-to-image models like DALL-E. Specifically, it first tokenizes a user's prompt, converting their words into a numerical vector. This vector guides the diffusion model, which iteratively reduces noise using the prompt to form a "coherent" image [14]. An image is then outputted after this process is complete. What differs Midjourney from other text-to-image models is that it seeks to focus on artistic expression and stylistic nuance, interpreting prompts with more flair and surreal visuals.

To create images, prompts are entered into Midjourney's website, located at https://www.midjourney.com/explore?tab=video_top. Once someone is subscribed, or using the free trial, they can enter a prompt. This prompt will create four images, where users can select the ones that best fit their desires. I entered each of the 20 prompts once, and saved three of the four produced images locally. I then uploaded these files to Github and my Google Drive outputs folder.

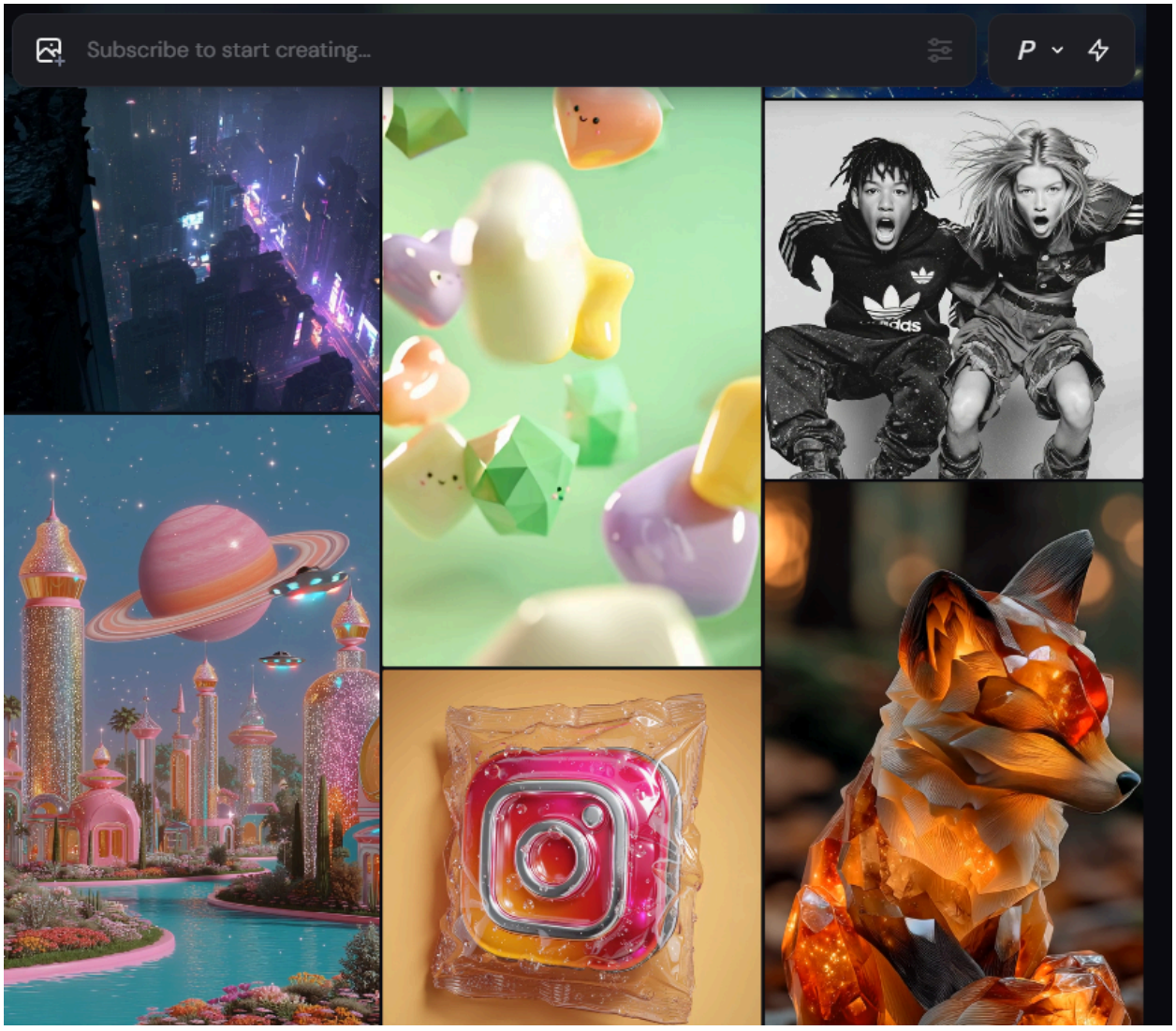


Figure [5]: Web Interface for Midjourney Image Generation

5.5.5 Firefly:

Adobe Firefly is a generative AI model developed with a great focus on ethical content generation and brand-safe design. It is trained exclusively on limited, licensed content in Adobe Stock [15]. This means that it has a generally smaller training dataset, but this dataset is all licensed and there are no worries regarding piracy and IP infringement. Although Adobe does not dissolve the complete model architecture, Firefly

is believed to work similarly to DALL-E, Midjourney, and other text-to-image models, being a diffusion-based generative model augmented with text encoders.

To generate images using Firefly, I went to Adobe's website located at https://www.adobe.com/products/firefly.html. I then entered each of the 20 prompts. Each entry generated four unique images, of which I saved three locally. I then uploaded these saved files to my Github repo and Google Drive outputs folder.

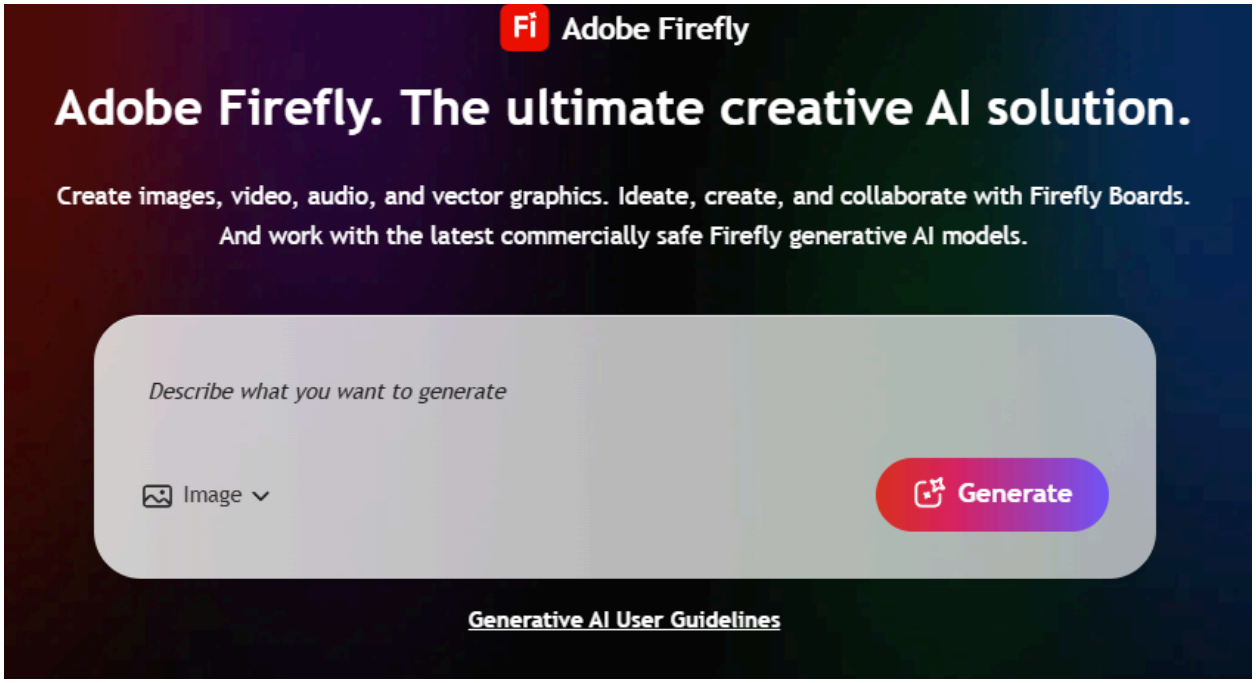


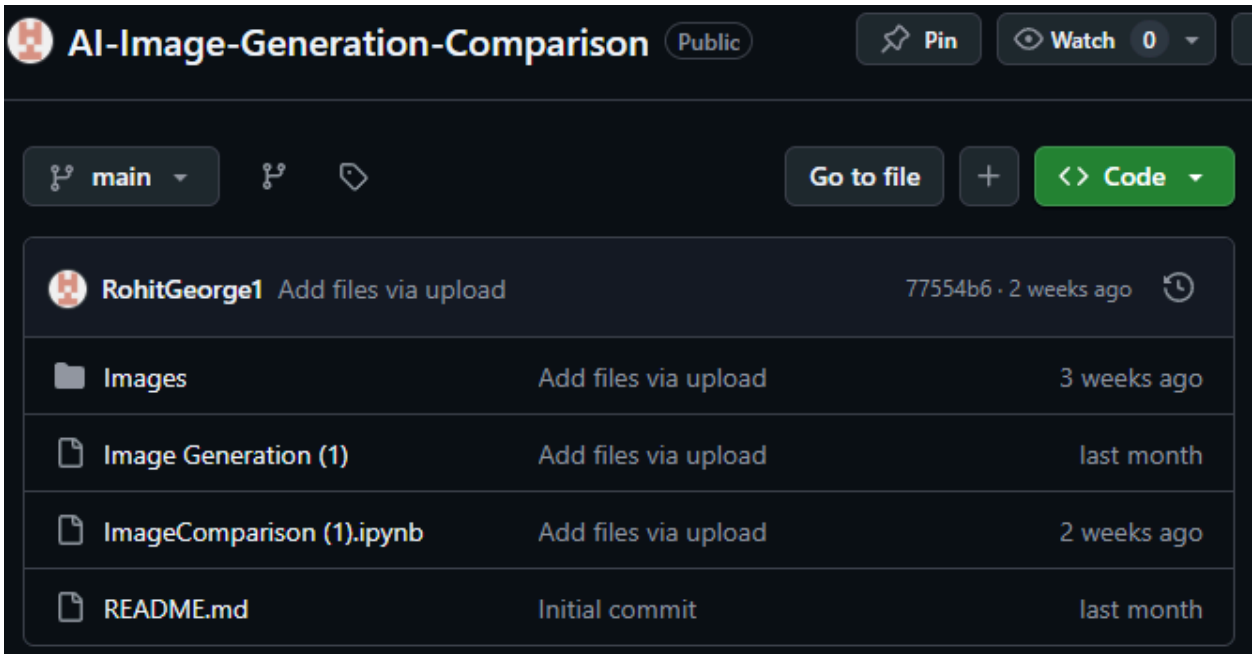
Figure [6]: Web Interface for Adobe Firefly Image Generation

5.5.6 Github:

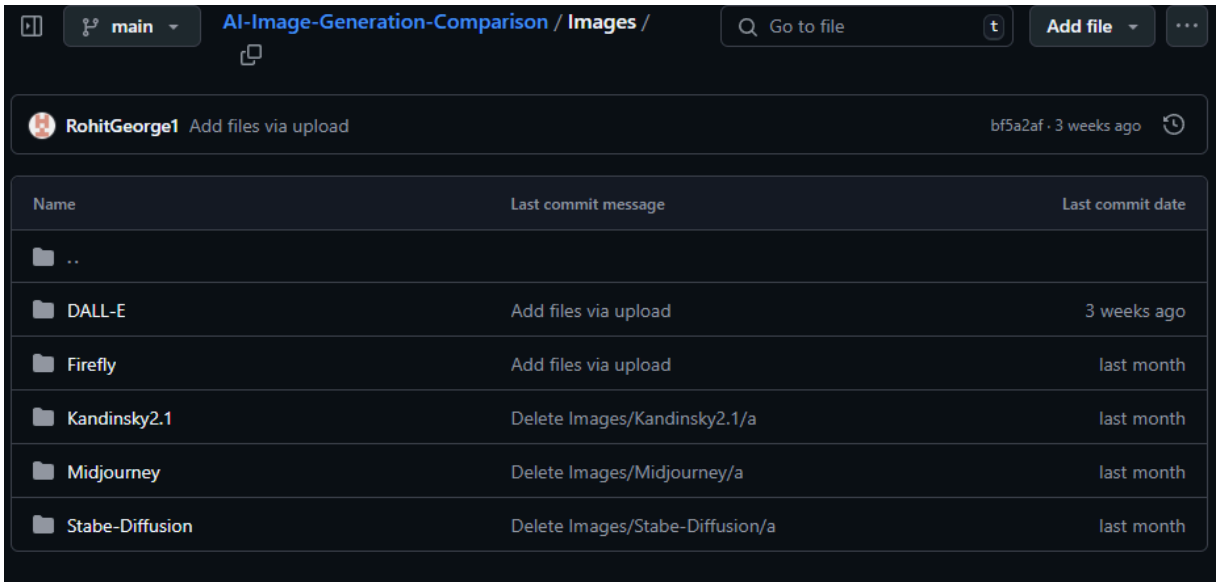
For version control and to allow others to easily view the code, all code was uploaded to github in a similar structure to the github storage.

A github repo called AI-Image-Generation-Comparison was created, with the url https://github.com/RohitGeorge1/AI-Image-Generation-Comparison.

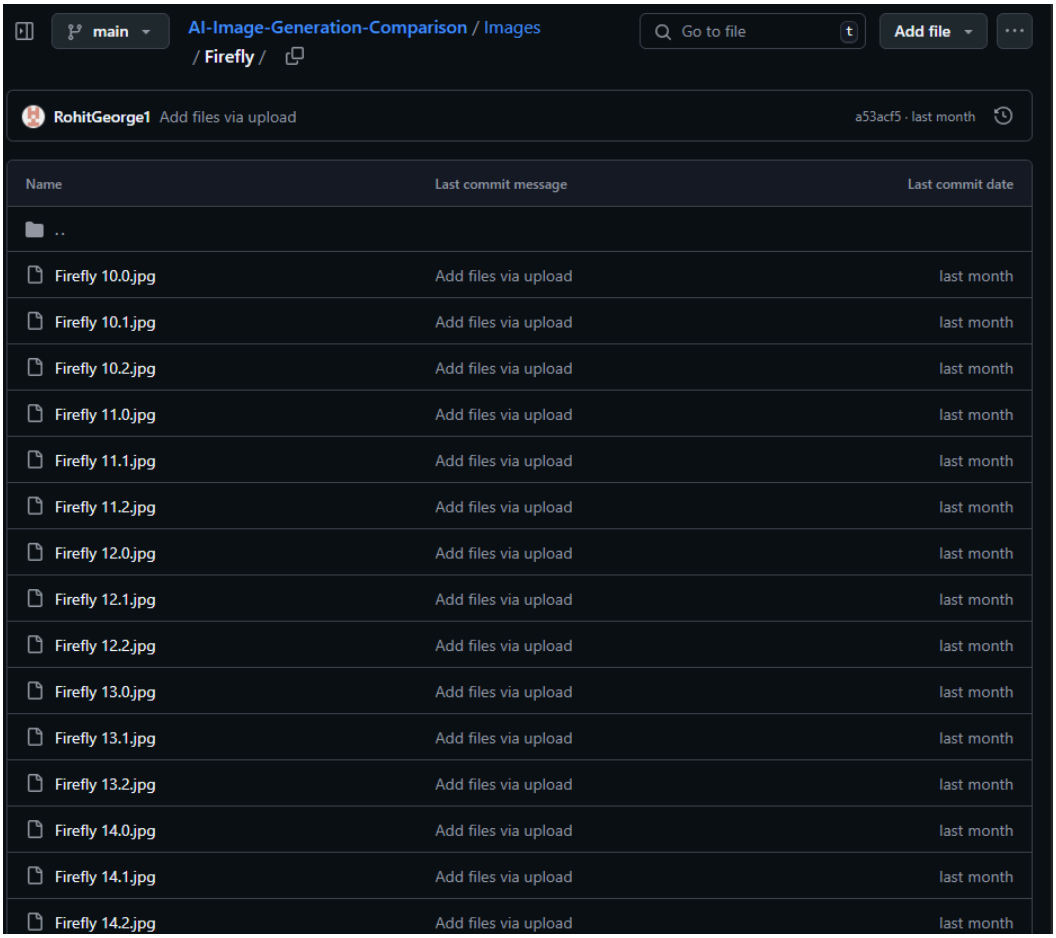
Inside this are an Images folder, Image Generation file used to generate images for Stable Diffusion and Kandinsky, ImageComparison file where all images across models and trials were compared, and a README.md file for setup instructions.



Inside the Images folder are folders for each of the models, where the 60 images generated for each of them are stored.



For example, AI-Image_Generation-Comparison/Images/Firefly/ looks as such:



5.6 Dataset Strengths and Limitations:

Strengths

- Controlled Prompting: Every model receives the exact same 20 prompts, ensuring equivalent, standard comparisons.

- Diverse Representation: The five models chosen span open-source, commercial, and artistic tools.

- Replicated Trials: Three generations per prompt allow intra-model variance and consistency checks.

- CLIP Compatible: Images are preprocessed for direct use with CLIP embeddings, t-SNE, PCA, and clustering.

- Flexible Reuse: The dataset structure allows for reuse in style analysis, real-vs-AI detection, fine-tuning models, or training classifiers.

Weaknesses:

- Scale: There are only 300 images, making the dataset small to medium sized. This may not be enough to support generalization across all model behaviors.

- Prompt Interpretation Logic: Each model has its own interpretation engine, meaning that same text may yield misaligned outputs. This is reviewed manually to ensure all prompt outputs make sense.

- Trial Randomness: Despite fixed prompts, stochasticity in generation can introduce variation.

- No real images: I don't have real images that correlate to prompts. This limits my ability to compare text-to-image models to real images. However, this project should still be able to aid people in telling AI generated images from real ones by finding trends that are generally found in text-to-image generated images.

5.6 Data Summary:

In summary, this dataset offers a well-controlled, diverse, and reproducible foundation for analyzing how leading generative models respond to identical prompts. Through consistent structuring, careful model selection, and built-in variability via multiple trials, it enables proper prompt-level and cross-model comparisons. It supports a wide range of analytic techniques, such as embedding based similarity clustering and latent space visualization, and addresses a critical; gap in understanding the semantic and stylistic behaviors of image generation systems. This dataset forms the backbone of the project's subsequent experiments and evaluations.

6. Implementation

6.1 Implementation Methodology:

The main methodology involved representing each image in a shared semantic space using CLIP (Contrastive Language-Impage Pretraining), a powerful vision-language model trained to align images and text in a common embedding space. Using CLIP allowed me to extract 512-dimensional embeddings for every generated image, thus allowing me to quantitatively compare visual outputs from the five different generative models. This is because CLIP encodes an image's generated features into a vector that captures pixel level information and high level concepts such as object, style, and scene structure. Trained on over 400 million text-to-image pairs, CLIP learns to understand how visual and textual content relate to one another, enabling the extraction of rich embeddings that reflect the image's underlying meaning and its alignment with the original prompt [16].

With these extracted embeddings, I applied dimensionality reduction techniques (PCA, t-SNE, and UMAP) to visualize clusters and latent similarities. I also used cosine

similarity to compute model-level and prompt-level distances. Prompt level analysis also allowed for deeper insight into how models responded to specific types of instructions, revealing differences in creativity and style along with biases and habits.

Overall, I compared the same prompt across different trials within a model, the same prompt across different models, and the overall latent similarity across models.

6.2 Implementation Setup:

The implementation was carried out entirely in Google Colab, using Python and a Jupyter notebook-based workflow. The notebook used was ImageComparison.ipynb, which included preprocessing, embedding extraction, dimensionality reduction, similarity comparison, visualization, and analysis code.

Key libraries included:

- CLIP from OpenAI's GitHub repository for extracting image embeddings
- Matplotlib and seaborn for plotting
- Scikit-learn for PCA, t-SNE, clustering, and similarity metrics
- Pandas and numpy for data handling
- Torch for running CLIP models on GPU

All generated image files were mounted via Google Drive (/content/drive) in a structure folder formal (/Outputs/ModelName/). The dataset of 300 images was processed in batches by iterating over directory contents and applying the CLIP preprocessing pipeline.

6.3 Implementation Steps:

- Environment Setup:
  The project was implemented using Google Colab as the main development environment. This included installing key libraries such as diffusers, transformers, matplotlib, sklearn, and CLIP. The folder structure was standardized and mounted via Google Drive to organize outputs by model. Each image was stored with consistent naming conventions to enable batch processing, evaluation, and visualization.

- Dataset Generation:
  The dataset was created by generating 300 AI-generated images using five different text-to-image models: DALL-E 3, Midjourney v6, Adobe Firefly, Stable Diffusion 1.5, and Kandinsky 2.1. Each of the 20 prompts was rendered three times per model to account for output variance. For the open-source models (Stable Diffusion and Kandinsky), generation was executed directly within Colab notebooks using the HuggingFace diffusers library. Closed-source models

required using web interfaces, with images being manually downloaded and renamed for consistency.

- Prompt and Trial Structuring:
  All prompts were categorized into five distinct themes to ensure stylistic and semantic diversity. Each generated image was labeled with its model, prompt index, and trial number. This uniform structure made it easier to parse results for intra-model and inter-model comparisons.

- Embedding Extraction with CLIP:
  To enable cross-model analysis, all images were processed using OpenAI's CLIP model (ViT-B/32) to extract 512-dimensional embeddings. This step allowed images to be represented in a shared vision-language embedding space, enabling comparisons not based on pixels but on the underlying semantics and visual structure.

- Preprocessing and Storage:
  Before embedding, all images were converted to RGB to meet CLIP's input requirements. A custom script iterated over each image file, extracted its embedding, and stored it in a structured Pandas DataFrame along with model name, filename, and promptID.

- Visualization and Clustering:
  Dimensionality reduction was applied using t-SNE and PCA to visualize relationships between images. Visual plots were created per prompt, showing how each model interprets the same prompt in high-dimensional space. KMeans clustering was then used on embeddings to uncover latent groupings and assess model level stylistic tendencies.

- Analysis Pipeline:
  The final analysis pipeline included distance-to-prompt calculations, intra-model variance, and cross-model divergence. Each result was output as a visualization, table, or plot, allowing viewers to understand model behaviour across prompts.

6.4 Major Blockers:

Early on, there were a few issues regarding accessing these models and uploading the images to my Google Colab output folder. For example, both Firefly and Midjourney were locked behind a paywall. I was able to access Midjourney by signing up for a free trial, but Firefly required me to spend $10. DALL-E also had a few issues. Originally, one could access DALL-E through an OpenAI API key. However, this feature

has been removed, forcing me to give up on generating the images in my ImageGeneration.ipynb file and do it manually in ChatGPT instead. However, this also resulted in a few issues as downloaded images were sometimes blank. Combined with the limited image production imposed by OpenAI, generating DALL-E images was very time consuming. Overall, Stable Diffusion and Kandinsky were automated to produce their respective images. However, having to generate Midjourney, Firefly, and DALL-E images manually limited the full automation of the data pipeline and took up much more time than the other two models.

Another issue was balancing the dataset size and the scope. With 300 images, deeper ML modeling such as classification was infeasible without the risk of overfitting, so the project focused on unsupervised analysis instead.

This issue also contributed to the similarity and uncertainty of outputs within certain models. For example, since I don't know how models view similarities in prompt categories, my prompt categories didn't always cluster as expected. Prompts with buildings, for example, were clustered together despite being initially placed in different categories. The reason behind outputs also contains some uncertainty, such as image variances causes.

7. Evaluation and Results

7.1 Global Comparison (All Models, All Prompts):

After extracting CLIP embeddings, I performed Principle Component Analysis (PCA) on the CLIP image embeddings, projecting them into a 2D graph. Points were colored by model and a global visual overview of how different model's outputs are distributed in CLIP's shared semantic space was produced.

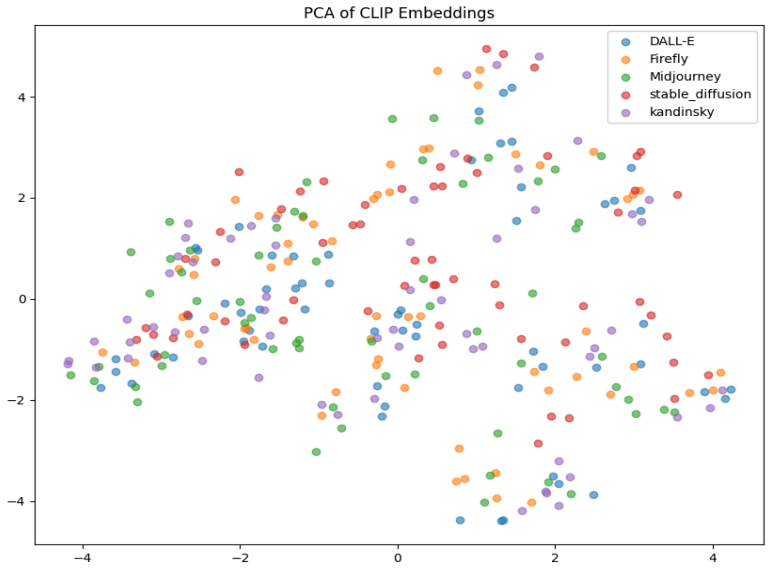
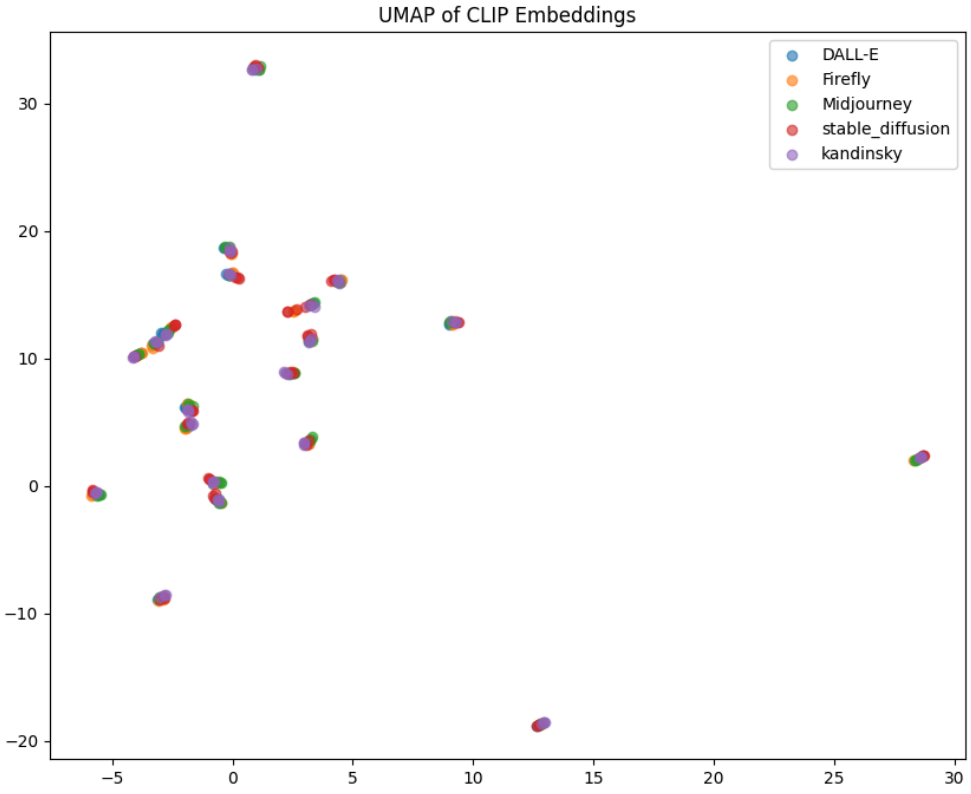


Figure [7]: PCA visualization of CLIP embeddings for all generated images. Each point represents a generated image, colored by the model that created it. The 512-dimensional CLIP embeddings are projected into 2D using PCA. Overlapping regions suggest similar semantic content across models. Separation highlights stylistic differences.

Overall, this first test wasn't too informative. I expected to see models grouped together or clusters based on prompts. Overall, the points are all over the place, regardless of model. This shows that all models can vary in production based on the prompt in order to meet prompt needs. Midjourney and Firefly seem to show the most spread, implying creative diversity or inconsistency, though no single cluster dominates. This shows that no model completely diverges from the rest as they share semantic prompts.

In addition to PCA, I applied Uniform Manifold Approximation and Projection (UMAP) to visualize the 512-dimensional CLIP embeddings in a 2D graph. I used this to identify consistent semantic groupings across models.



Figure[8]: UMAP projection of CLIP embeddings, colored by model. Each group of 5 overlapping points represents the same prompt rendered by different models.

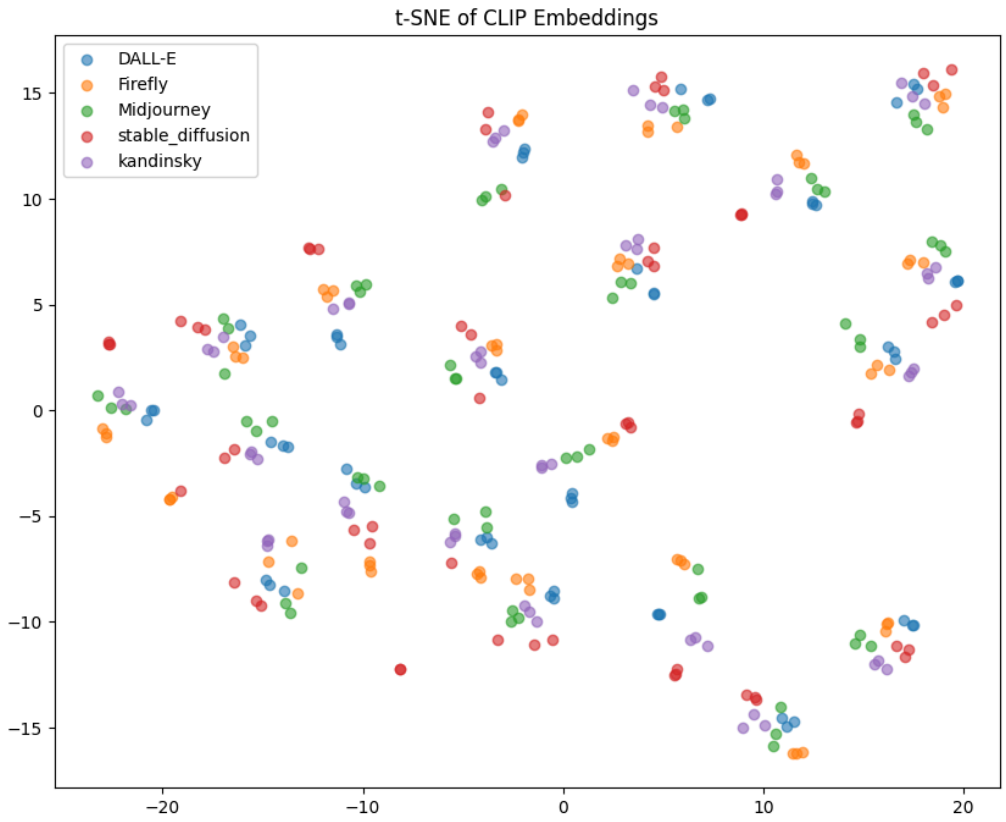
As seen in the above UMAP figure, each prompt appears to form a tight cluster, with all five models densely overlapping at each prompt point. This indicates a large

amount of inter-model agreement at the semantic level, suggesting that models tend to agree in their core interpretation of prompts. There is a large amount of semantic consistency that despite stylistic differences, the core interpretation of the text prompt always carries the most weight and leads to all of these models having tight semantic consistency. As such, this supports the conclusion that CLIP embeddings reliably encode semantic meaning in a manner that isn't dependent on the model used and that prompt identity is the dominant driver of latent space structure.

7.1.2 Model Clustering Behavior:

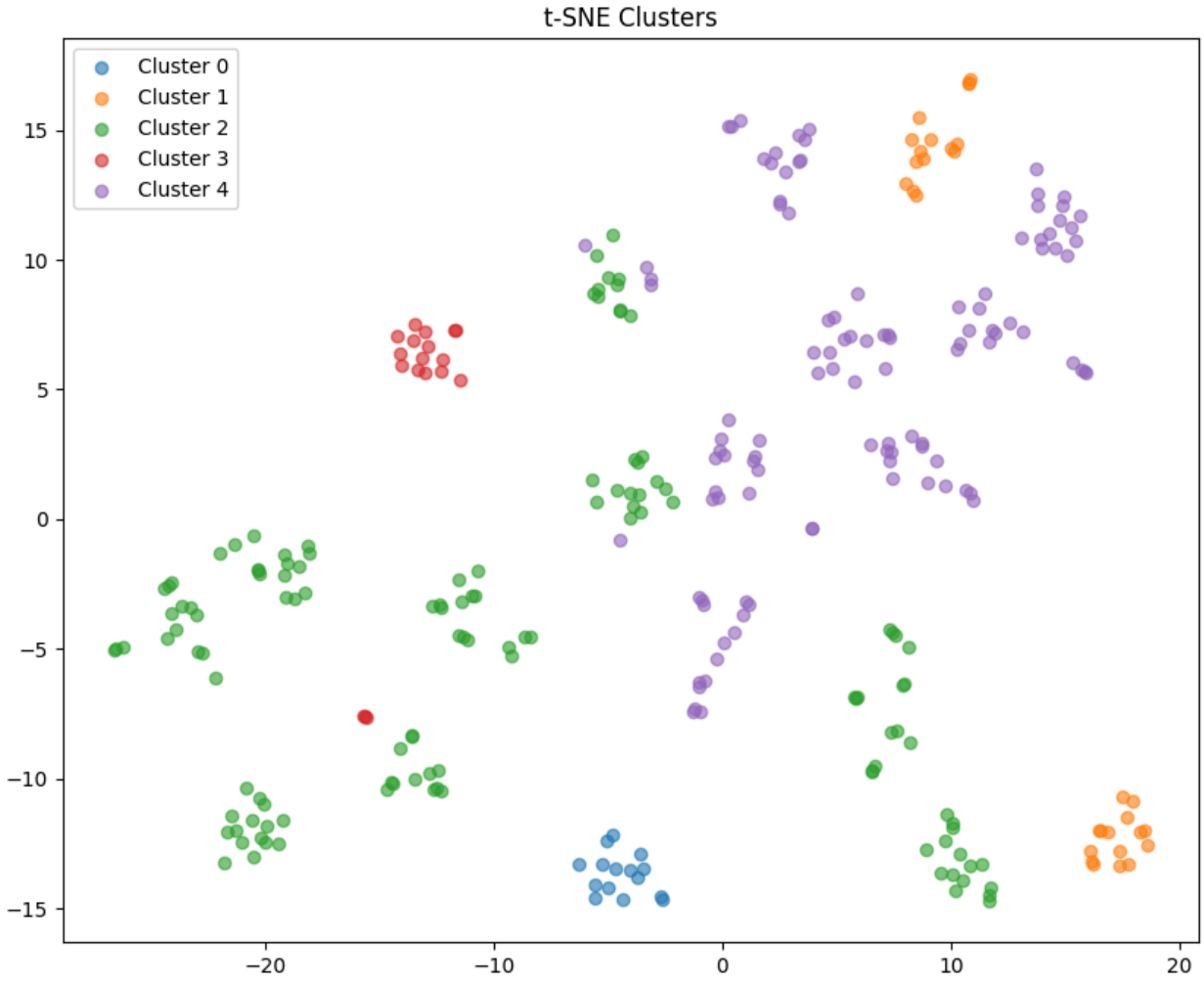
I then used CLIP embeddings to perform t-SNE to further understand the stylistic and conceptual similarities between image generations from different models. I used CLIP embeddings to project each image into a shared semantic space. I then used t-SNE to visualize these embeddings. t-SNE is a dimensionality reduction technique that preserves local structure in high-dimensional data.

Figure [9] shows a t-SNE projection colored by model. This visualization highlighted that no model was completely isolated. This suggests that all models share some visual-semantic space. Visually, DALL-E and Firefly seemed to form tighter local clusters, indicating a greater consistency in their outputs. We can also see a few clusters taking place naturally, most likely a result of different prompts and prompt categories coming together.



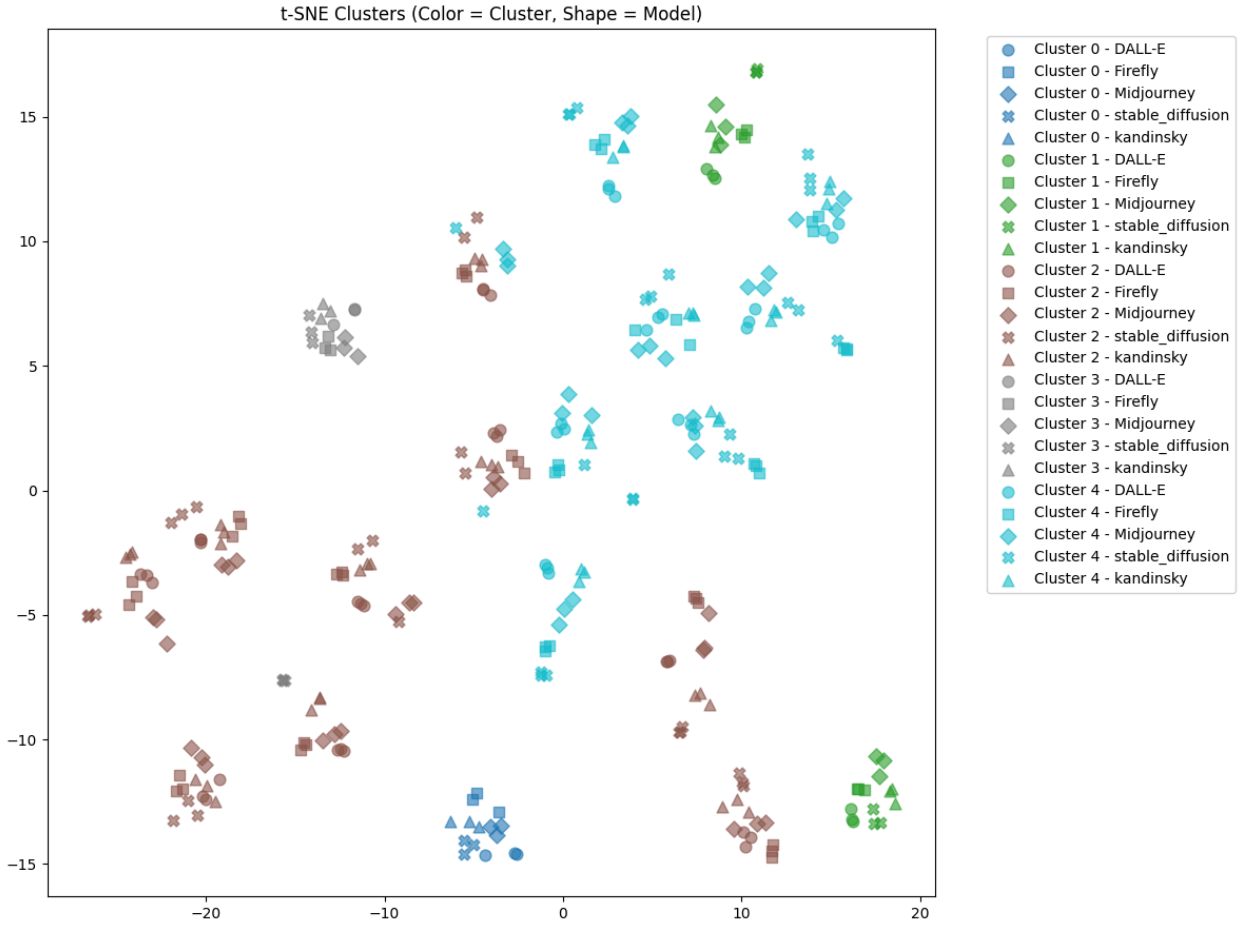
Figure[9]: CLIP embeddings projected with t-SNE to display CLIP embeddings in 2D form.

Building off of this, I applied KMeans clustering to the CLIP embeddings and projected the cluster assignments onto the t-SNE space, where clusters represented the prompt category, resulting in 5 total clusters. This is seen in Figure[10]. From this, we see that different prompt types result in very different clusters. Animals in Unusual Contexts, Urban + Nature Blend, and Human + Sci-fi resulted in densely packed clusters. However, Human + Sci-fi did have one outlier. Meanwhile, Style comparison and Surreal were much more diverse. However, as categories, they are also much broader. We can still see clusters forming based on specific prompts, seeing roughly 15 images close together for each prompt. Another thing we can see is some clusters bleed into other clusters, suggesting similarities in results despite prompt differences. This shows that some concepts (Human + Sci-Fi and Style Comparison for example) had similarities in creation despite different prompts.



Figure[10]: KMeans applied to CLIP embeddings and displayed with t-SNE to help show clustering within prompt categories.

Since this image didn't reveal what models were producing which node/dot, I created a new image seen in Figure [11], which showed which models produced which node. Overall, this helped aid the conclusion that groupings within clusters largely arose due to the same prompts, suggesting little variance in prompt creation and supporting the idea that the prompt content itself plays the largest role in the final image, regardless of model used. We can also see that the outlier in cluster 2 was Stable Diffusion, suggesting that there are larger variances in its results and possibly suggesting worse performance.



Figure[11]: CLIP embeddings clustered and visualized in 2D using t-SNE, along with the models that created each node.

Prompt Distribution by Cluster:

prompt_id	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	\
cluster																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	15	0	0	0	15	0	0	0	0	0	0	0	
2	15	15	14	15	0	0	0	0	15	0	11	15	0	0	0	15	12	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	
4	0	0	1	0	15	0	15	15	0	0	4	0	15	15	15	0	0	

prompt_id	17	18	19
cluster			
0	15	0	0
1	0	0	0
2	0	0	0
3	0	0	15
4	0	15	0

Figure[12]: Table that displays the clusters each prompt ID was clustered into.

Contrary to my belief that clusters were formed based on prompt topic, we see that many prompt groupings, such as prompts (16-19, Surreal) are split into many different clusters. There are a few explanations for this. For one, models saw different groupings than the ones I proposed. For example, prompts 4,6,7, and 18 all revolve around landmarks and buildings, and are thus likely in the same cluster. However, other prompts are in the same cluster yet focus more on people (prompts 12 and 13). As such, this could also support the conclusion that model-specific styles and visual execution still play a major role in prompt results, rather than just the prompt content itself. Supporting this conclusion are prompts 2, 10, and 16, which differ in clusters despite having the same prompts. As such, it can be concluded that same prompts will result in different image outputs, despite having consistently similar embeddings as shown by UMAP. Overall, 17 of the 20 prompts had all of their respective images in the same cluster, suggesting that prompt content is still the most important determiner of the image output. This also suggests that every model has a similar encoding structure and generates similar embeddings. However, these are not complete matches as there are still variances in the final output, sometimes to a large degree.

7.1.3 Average Prompt Similarity Across Models:

To compare how semantically aligned different models are on a per-prompt basis, I computed the mean CLIP embedding for each model across its three trials for each prompt. These averaged embeddings were then projected into 2D using UMAP with cosine distance (Figure [13]).

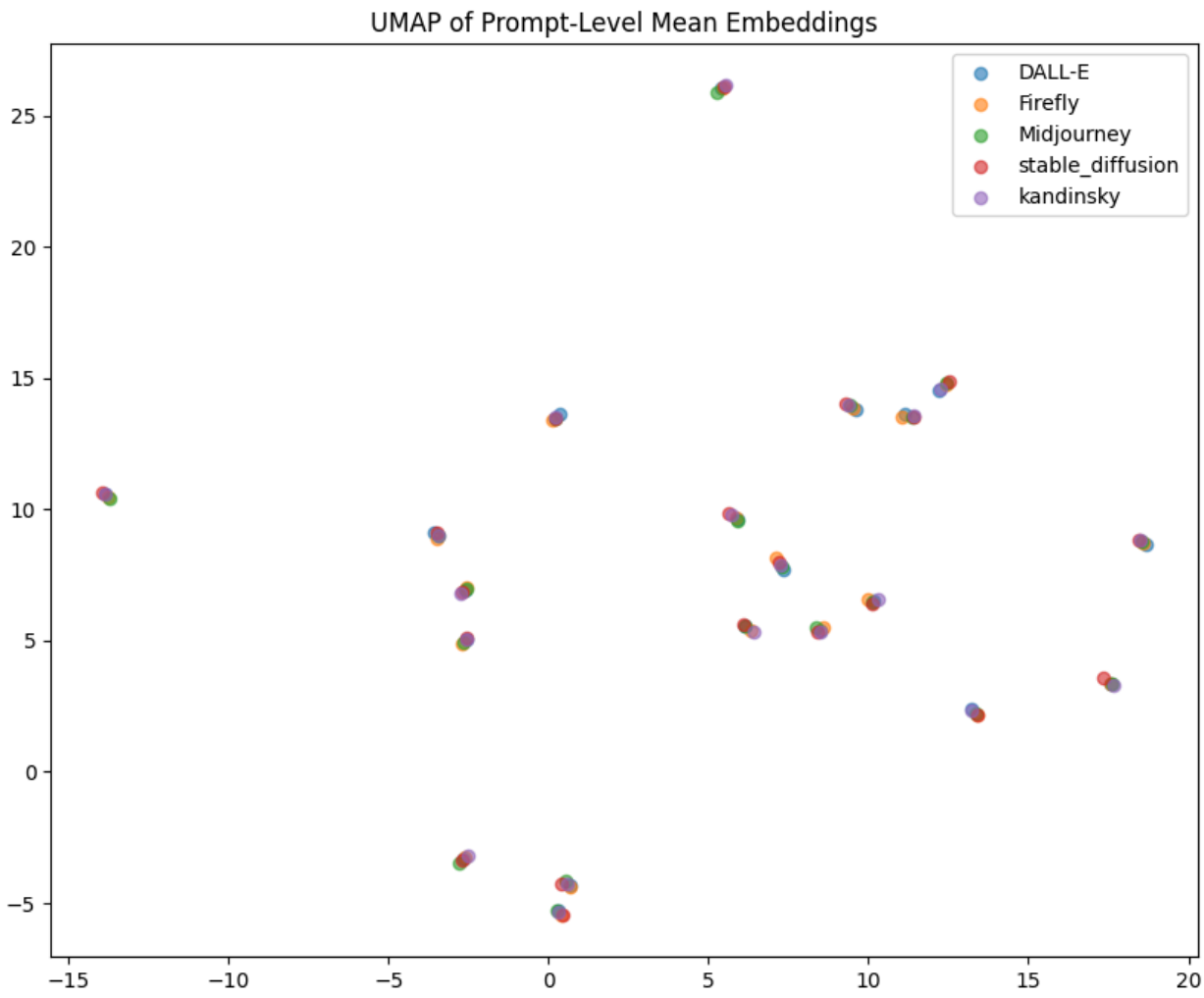


Figure [13]: UMAP projection of mean CLIP embeddings for each model per prompt (20 prompts per 5 modals for a total of 100 points). Each group of 5 points represents one prompt. The tight grouping of each prompt cluster suggests strong cross-model semantic agreement.

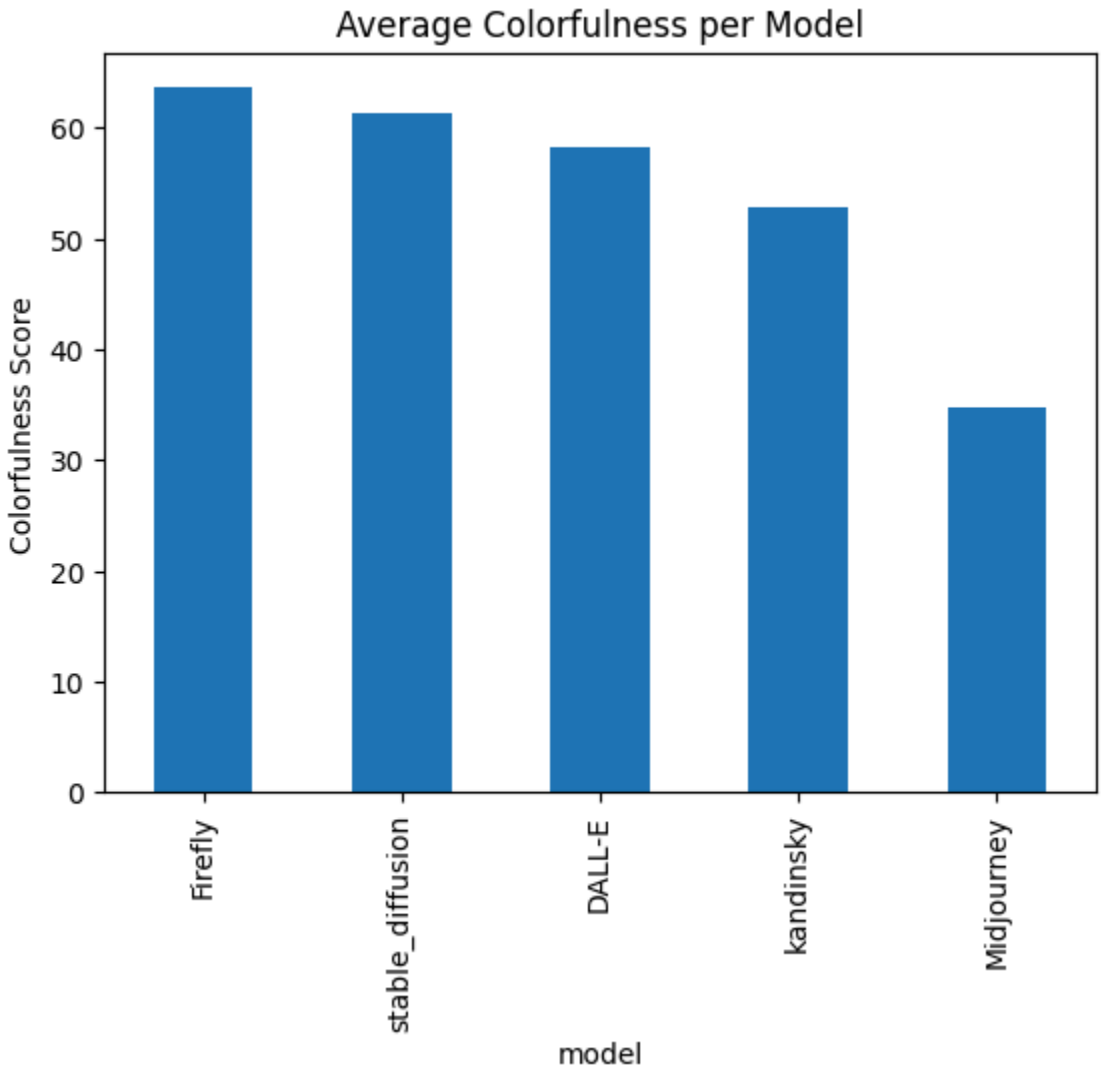
Each group of closely packed points represents that same prompt, rendered by each of the five models. The consistent tightness of these prompt clusters indicate that despite stylistic and architectural differences, the models share a strong semantic agreement in their interpretation of the prompts.

This supports the conclusion that models interpret prompts similarly and develop their resulting embeddings in a similar manner to one another. There are small deviations, but the start of the image generation process (tokenizing the prompt and generating embeddings) is similar across models. Resulting differences thus likely take place in later steps of the process.

model	DALL-E	Firefly	Midjourney	kandinsky	stable_diffusion
model					
DALL-E	1.000000	0.926642	0.960333	0.957665	0.930079
Firefly	0.926642	1.000000	0.955414	0.971290	0.958700
Midjourney	0.960333	0.955414	0.999999	0.978275	0.942862
kandinsky	0.957665	0.971290	0.978275	1.000000	0.966327
stable_diffusion	0.930079	0.958700	0.942862	0.966327	1.000000

Figure[14]: Cosine Similarities between mean embeddings. 1.0 means a perfect match.

Performing cosine similarities on these mean embeddings showed that they were indeed highly correlated, supporting the earlier paragraph.



Figure[15]: The average colorfulness score of each generation image model. Higher scores indicate more vibrant and varied color usage.

Finally, I extracted the RGB channels of each image using OpenCV and PIL and computed their colorfulness score using the metric created by Hasler and Susstrunk [17]. Firefly had the highest average colorfulness score, suggesting that it tends to produce the most vibrant and saturated images. Stable Diffusion DALL-E, and Kandinsky followed closely in that order. Midjourney had the lowest colorfulness score, implying it uses less color than the other models. Overall, this reinforces the conclusion that the models follow similar steps through the embedding process. However, when they start to reduce noise, they have different tendencies regarding colors and other details that reflect their unique styles.

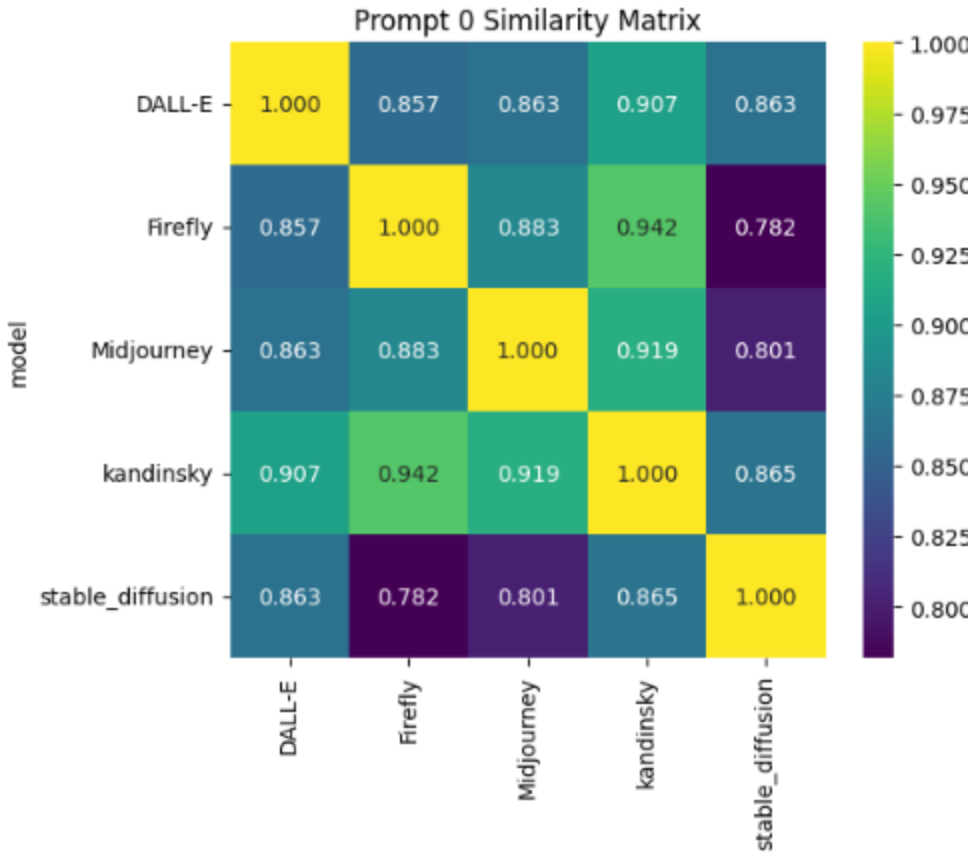
7.2 Inter-Model Comparison (Same Prompt, Different Models)

7.2.1 Embedding Similarity Scores:

To compare models across the same prompt, I began by taking the prompt-level average embeddings per model. This was done by computing the mean embeddings across three trials for each image. This gave a single embedding vector per model for each prompt. For each of these prompts, for every pair of models, such as DALL-E and Midjourney, I calculated the cosine similarity between their average embeddings. This gave me the average model similarity across all prompts, showing how often two models tend to produce similar outputs when given the same prompt. As seen in Figure [16], all of these models are quite similar in how they output semantic content based on a prompt, sitting between 0.8 and 0.9 across all pairings. Some models, such as DALL-E, Kandinsky, and Midjourney, are more alike in their semantic outputs, having a greater than 0.88 similarity score. Meanwhile, Stable Diffusion is more different from the others, having the lowest similarity score generally. Still, all of these are quite high. It should also be considered that even within the same model, embeddings will not be the same across trials of the same prompt. This is explored later.

	model1	model2	similarity
0	DALL-E	Firefly	0.853031
1	DALL-E	Midjourney	0.885129
2	DALL-E	kandinsky	0.886584
3	DALL-E	stable_diffusion	0.815420
4	Firefly	Midjourney	0.858032
5	Firefly	kandinsky	0.869942
6	Firefly	stable_diffusion	0.832362
7	Midjourney	kandinsky	0.888418
8	Midjourney	stable_diffusion	0.819312
9	stable_diffusion	kandinsky	0.853606

Figure[16]: Average CLIP embedding similarity scores between all models, where 1.0 represents exact matches.



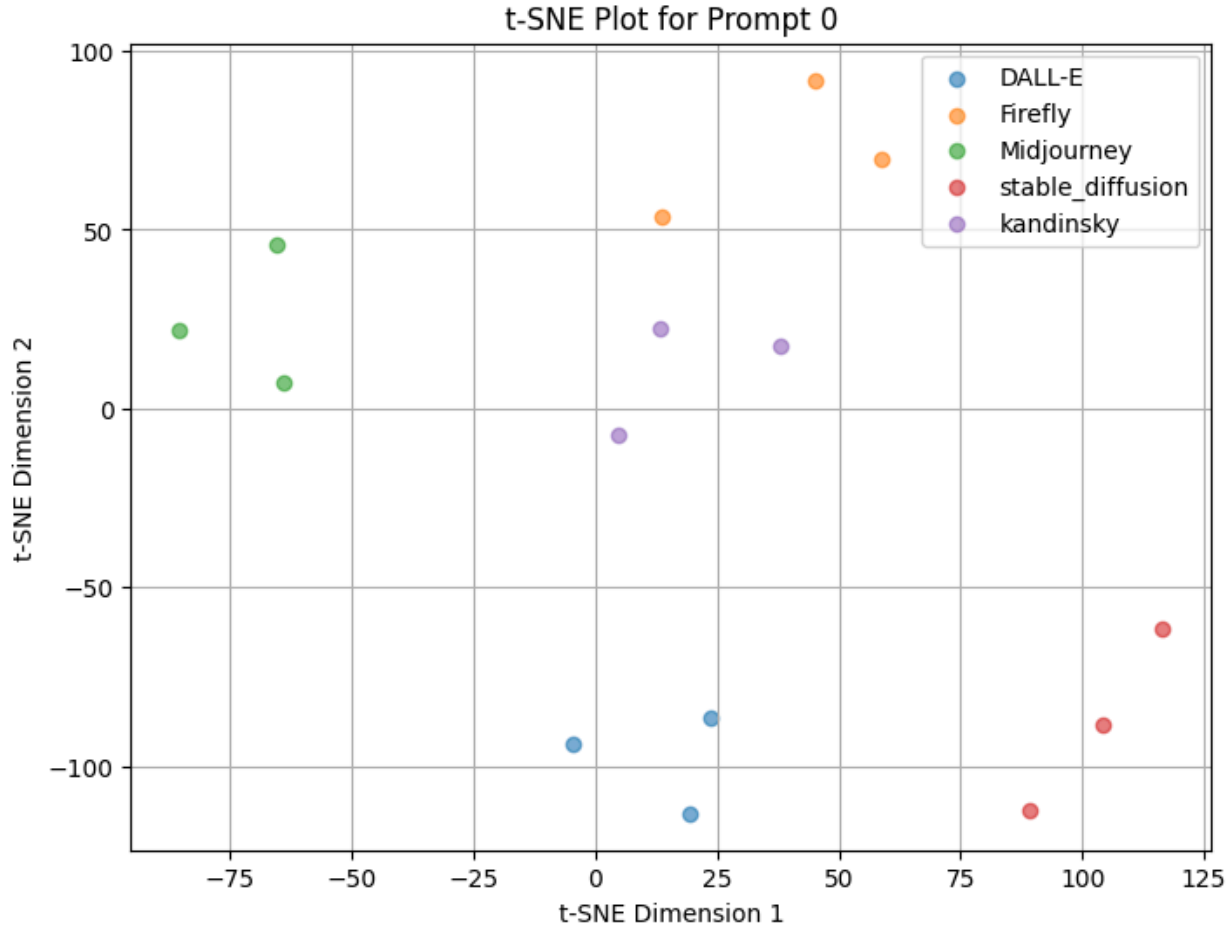
Figure[17]: Similarity Matrix Containing Average Similarity Scores For Each Model For Prompt 0.

As seen in Figure [17], I then proceeded to create heat maps displaying the similarity scores of the mean embeddings per model for each prompt. Overall, these displayed very high similarity scores. In general, the lowest scores for each prompt were still greater than 0.75. Meanwhile, there were many values greater than 0.8, with quite a few even exceeding 0.9. This once again highlights the large similarity between these models in the text embedding stage. Since it is known that Stable Diffusion uses a CLIP for extracting text embeddings from a prompt, it is likely that all models use a CLIP model for extraction. However the exact model varies between them, with some being more similar than others.

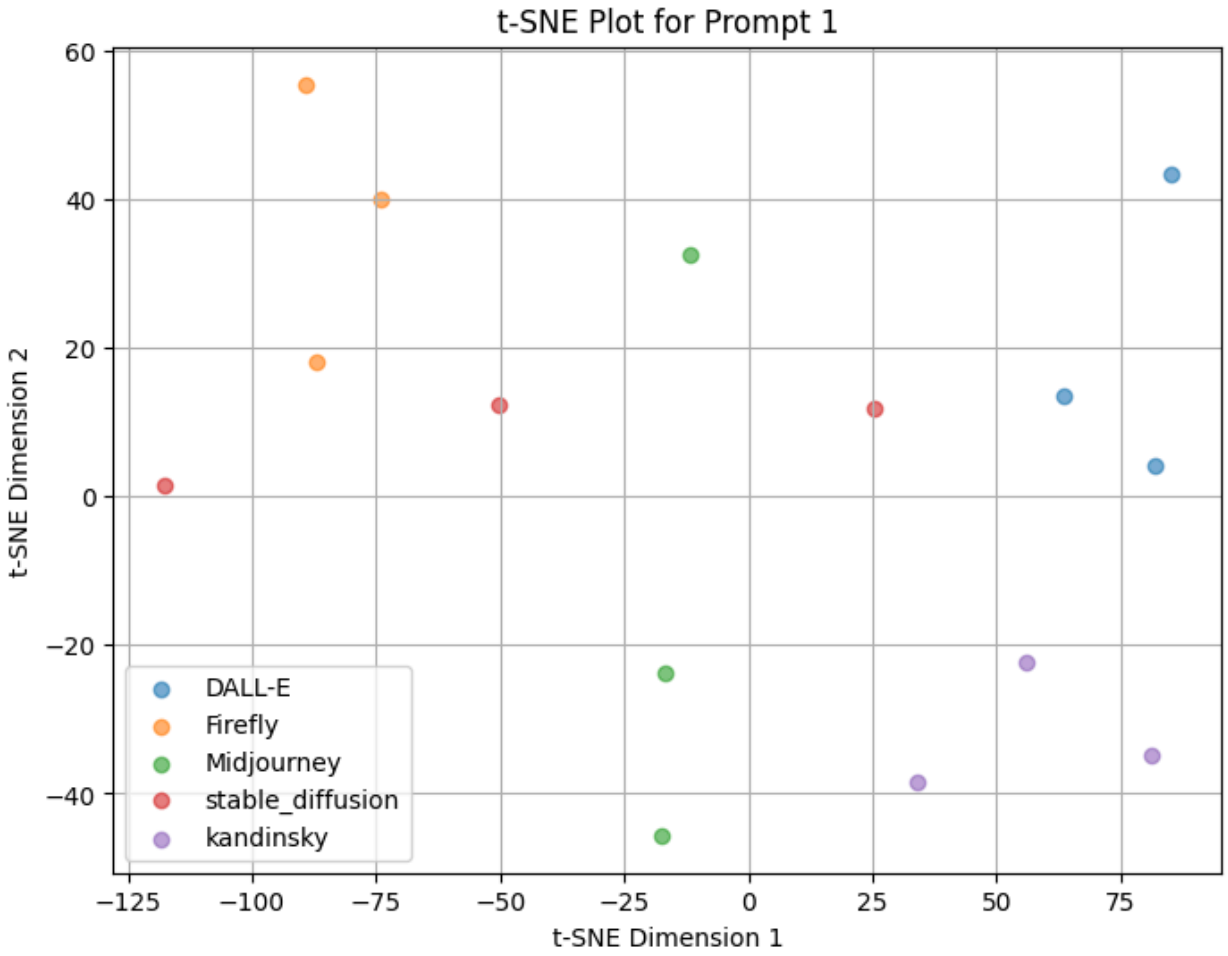
7.2.2 Per-Prompt Embedding Maps:

Continuing to analyze embedding variances and similarities, I created t-SNE plots displaying the embeddings of each trial for each model in each prompt. Figure[18] shows this for Prompt 0. This image highlights the difference between Stable Diffusion and the other models, as Stable Diffusion is more isolated in the graph. Furthermore, we see that the embeddings within models form more of a cluster. This makes sense as these embeddings would be created through the same text extraction model. This

suggests that intra-model images hold more similarity in their embeddings than in their embeddings with other models, which makes sense. However, while this was true for the majority, this was not true across the board. Figure [19] displays a large amount of variance for Stable Diffusion in prompt 1. This caused one trial to correlate more with those of other models, specifically those of DALL-E and Kandinsky. These cases were repeated a few times, suggesting two things. For one, Stable Diffusion likely sees the largest variance in its produced images among the tested model, a conclusion supported later on. Secondly, the conclusion that the embedding extraction model used is similar is once again supported, as embeddings vary within a model's own trials and can be very similar to those of other models.



Figure[18]: t-SNE Plot Displaying Embeddings For All Models Across All Trials for Prompt 0. Each dot is a different image for the same prompt. Colors are based on the model used for production. Widely spaced dots mean more semantic variation in how different images interpreted the prompt.



Figure[19]: t-SNE Plot Displaying Embeddings For All Models Across All Trials for Prompt 1. Each dot is a different image for the same prompt. Colors are based on the model used for production. Widely spaced dots mean more semantic variation in how different images interpreted the prompt.

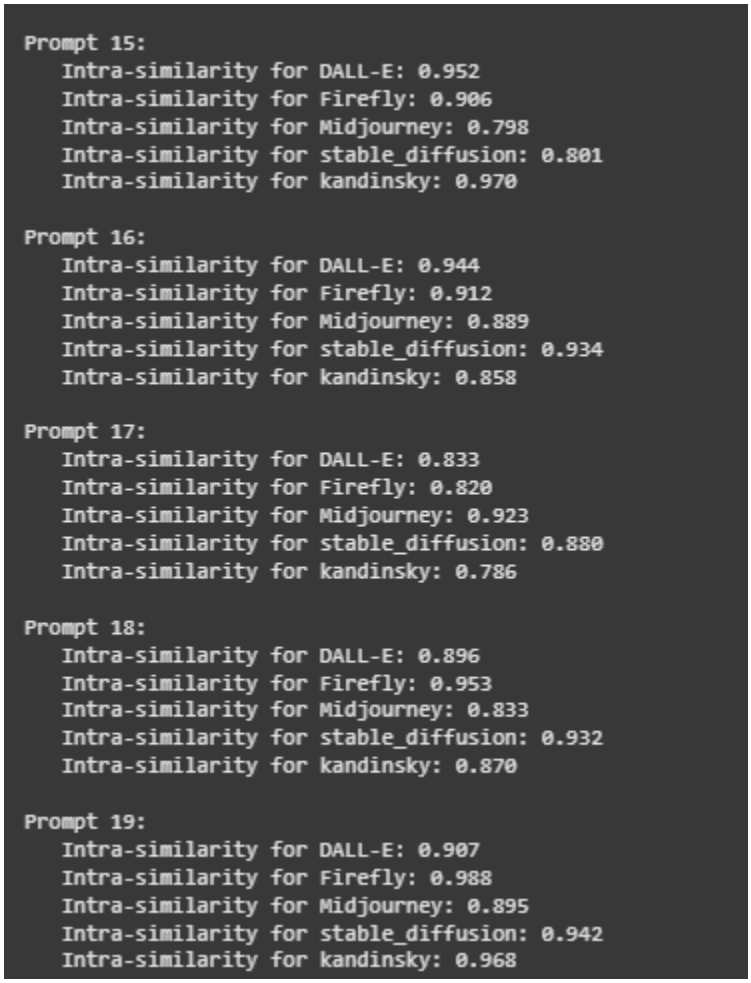
7.3 Intra-Model Comparison (Same Model, Same Prompt, Multiple Trials)

7.3.1 Trial Consistency Scores:

I then chose to look within the trials of each prompt for each model to determine how consistent they were. This would provide insight into model consistency, model adherence to the prompt, and model creativity.

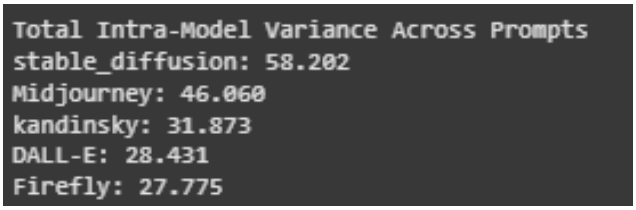
Figure [20] and Figure [21] measure the consistency of each model across multiple generations of the same prompt (using intra-prompt CLIP similarity). Figure [20] took the average pairwise CLIP cosine similarity between the trials of each prompt. A high intra-similarity meant that the model gave nearly identical outputs, with 1.0 meaning fully identical. This indicates high consistency and low diversity. Low

intra-similarity would indicate that a model gives different outputs each time, suggesting that it is quite creative.



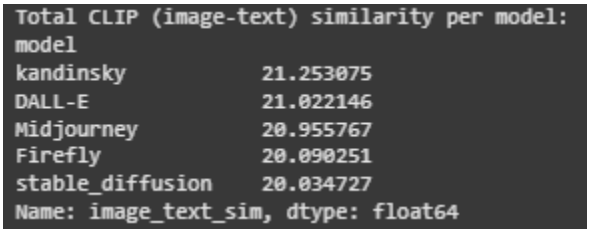
Figure[20]: Average Pairwise Cosine Similarity Across all Trials For Each Model For Each Prompt

Figure [21] measured the variance of intra-prompt CLIP similarity, measuring how consistent each model is across multiple generations of the same prompt. A higher value meant more variation in outputs for the same prompt. This can represent creativity and diversity. This shows that Stable Diffusion displayed the greatest variance in its outputs, followed by Midjourney. Kandinsky, DALL-E, and Firefly are all similar in their variances and are much less than that of Stable Diffusion. This suggests that Stable Diffusion may favor more open-ended or creative generation behaviors. Meanwhile, Firefly, DALL-E, and Midjourney seem to have more deterministic and consistent outputs.



Figure[21]: Total Intra-Model Variance Across Prompts. Variance Is Measured Across Trials For Each Prompt and Summed For Each Model.

I then summed the CLIP image-text similarity scores across all generated images per model. Specifically, for each prompt and generated image pair, I computed the cosine similarity between the image embedding and the corresponding text prompt embedding using the CLIP model. These per-image similarity scores quantify how closely the visual content of each image is aligned with its text prompt in CLIP's joint embedding space. By grouping the results by model and summing these similarity scores across all prompts and generations, I obtained an aggregated measure of total prompt-image alignment. This is seen in Figure [22]. A higher total indicates that the model consistently produces images that are semantically close to their prompts across the dataset, as judged by CLIP.

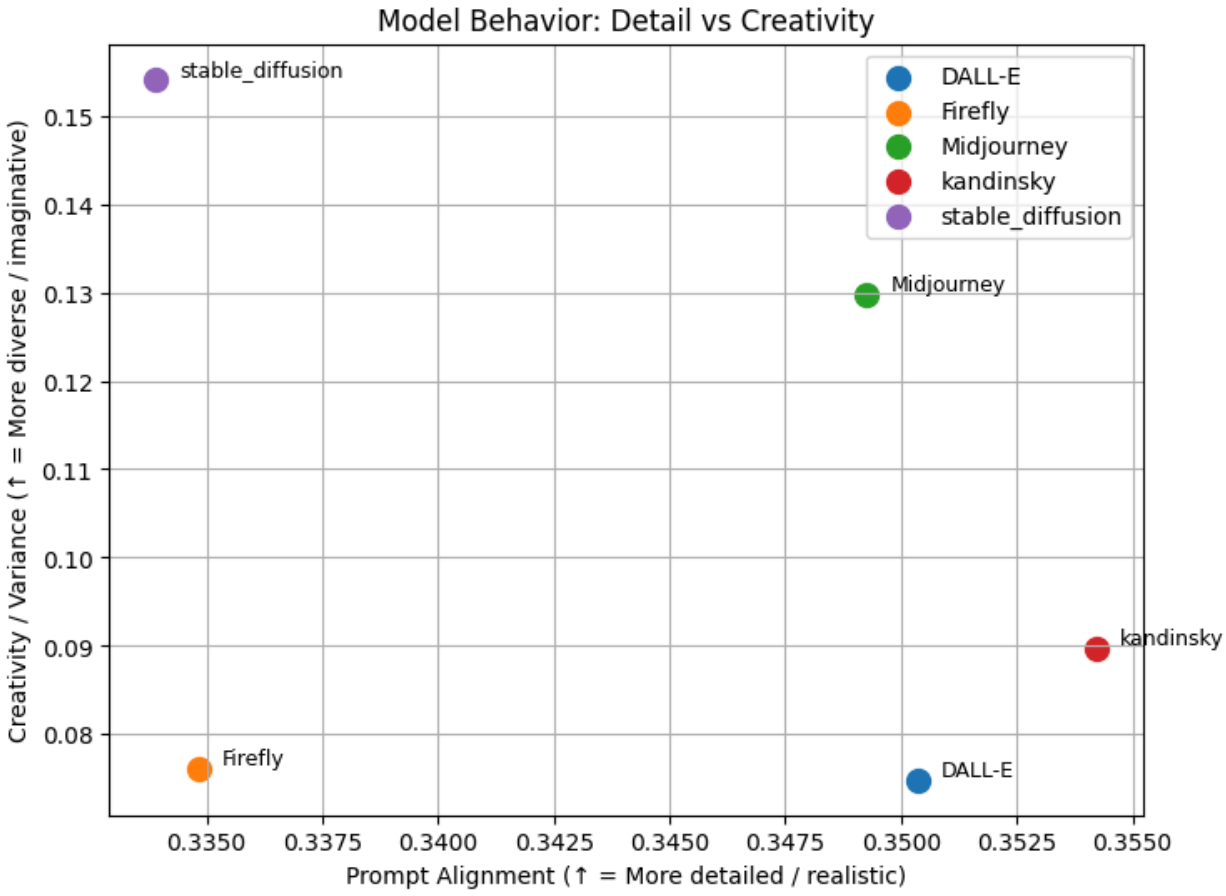


Figure[22]: CLIP Similarities Between The Generated Image Embedding And Text Prompt Embedding Summed For Each Model

These results tell me that all models are quite similar in how their image embeddings match the text embeddings of the prompt, supporting the conclusion that their text extraction and embedding process is similar. Overall, Kandinsky and DALL-E had the best prompt alignment, suggesting a higher fidelity to the input prompt and stronger adherence to prompt semantics. Stable Diffusion had the lowest alignment, indicating its outputs are less tied to the prompt.

These can go hand in hand in some cases. For example, Stable Diffusion had the lowest image-text alignment while it was the highest in variability, suggesting there may have been a trade off. This led me to combine the two, creating Figure[23]. This figure models creativity and variance against prompt alignment. We see that DALL-E and Kandinsky exceed in prompt alignment while they suffer in variance, suggesting they are stronger in prompt adherence and consistent image production. Stable

Diffusion, as mentioned, is the opposite, favoring variability over strong prompt adherence. Midjourney is a good mix of both, while Firefly is very low in both categories.



Figure[23]: Graph of Prompt Alignment Sum on the X-Axis vs Variance Sum on the Y-Axis For Each Model.

8. Conclusion

This work set out to compare and contrast five popular generative model models: DALL-E, Midjourney, Firefly, Stable Diffusion, and Kandinsky. To do this, their outputs were embedded in CLIP's joint image-text space. Through a series of dimensionality reduction techniques, clustering algorithms, and similarity metrics, I was able to uncover differences in how each model semantically interprets and outputs the same text prompts.

One of the most consistent findings across the produced visualizations was a strong agreement across all models at the semantic level. Despite any differences in architecture, style, or diffusion, all five models produced similar embeddings for each prompt. This was seen clearly in the UMAP projections, where each prompt formed

tightly knit clusters. This suggests that prompt content is the dominant factor shaping the semantic embedding space and that all models use a similar CLIP model to retrieve text embeddings from the prompt and form similar image embeddings in response.

However, stylistic and creative differences still emerged, particularly in intra-model variability and prompt-image alignment. Stable Diffusion had highest intra-model variance, meaning that it produced the most diverse outputs across trials of the same prompt, thus indicating that it has great creative variability. However, it also had the lowest image-text alignment sum, suggesting that its outputs were less faithful to the original prompts. On the other hand, DALL-E and Kandinsky produced highly consistent and prompt-aligned outputs, having the highest image-text alignment sum, while also having lower creative variance. Midjourney seems to be a balanced performing, achieving both moderate creativity and alignment. Firefly was low in both fields, indicating a more constrained generation strategy. This is likely due to its limited dataset, with it only being trained on Adobe Stock[].

As such, users who want more variable and creative images would likely want to use Stable Diffusion to produce their images. Meanwhile, those who desire consistency and strong adherence to their prompt with limited creative input would likely have the best experience using DALL-E and Kandinsky. Those who would like a mix of both would choose to use Midjourney. Adobe Firefly was the most colorful, but performed poorly in both image variance and prompt fidelity. Users would use it if they desired colorful images or if they care more about IP infringement and the training dataset used by the model.

Overall, I found that the prompt itself is the primary driver of the final image output across models, quite understandably given the nature of prompt-based generation. Furthermore, the text embedding and image embedding steps appear to be very similar across models. This is likely due to their shared reliance on CLIP or similar architectures. However, the denoising and decoding stages seem to have larger differences, leading to lower cosine similarities (~0.8-0.9) than identical models would have (~0.95+). These differences likely stem from variations in training datasets, model architectures, and VAEs. Despite this, the models are still extremely similar, exhibiting a large degree of semantic similarity in how they interpret and represent prompts.

From a methodological perspective, this project also validates the use of CLIP embeddings as a powerful, unified lens for analyzing and comparing generative models. Since CLIP can embed both text and images into the same latent space, it serves as an effective intermediary for measuring prompt-image alignment, model consistency, and semantic clustering across diverse architectures.

CLIP similarities across models also highlights the ability for real images to be different from model-generated images. Since all the tested generated models performed similarly in latent space embeddings, one can test an image by extracting its embeddings and comparing them to the embeddings of similar model-generated

images. A high cosine similarity to any of these models would indicate that the image was model-generated.

While my current work represents just the beginning, it establishes a strong foundation. The results so far demonstrate that even with a modest dataset, meaningful semantic differences between models can be uncovered and visualized. This paces the way for more scalable, interpretable, and user-friendly tools for navigating the rapidly evolving space of generative image models. With a larger and more varied dataset, one could refine these analyses to develop automated model fingerprinting, perform fake image detection based on embedding similarity, or even build interactive tools that help users navigate the generative space based on alignment, creativity, or style. The techniques demonstrated in this paper scale well, are interpretable, and offer a foundation for understanding the rapidly expanding ecosystem of AI image generation.

9. Future Work:

There is much more work that can be done. One of the most impactful next steps would be to greatly expand the dataset, increasing the number of prompts from 20 to around 50-100. More categories could also be added. This would improve the generalizability of the findings and offer a more robust assessment of each of the model's strengths and weaknesses across different semantic domains. A larger dataset would also support more sophisticated machine learning techniques, such as classification algorithms, to identify the source model from an image. With more data, models would be better able to capture small stylistic or structural patterns across different models. This would help advance one of the motivations of this paper, allowing people to tell where images come from and whether they are real or model generated.

To build off of this, incorporating real images as baseline comparisons would ground the analysis in natural image statistics and offer insight into how "realistic" each model truly is in embedding space. Including these images would allow classification machine learning models to better determine which images are real and which are not.

Similarly, more models could be incorporated, such as Leonardo AI, Canvas, and Gemini, along with newer versions to both capture progress in this field and determine more trends in image generation models.

Even without a machine learning algorithm, one can test whether an image is real by generating similar images using these models and computing the cosine similarity score of the tested image to those generated. A high cosine similarity score will indicate that the image is likely AI-generated.

Furthermore, style embeddings or fine-tuned CLIP could be used to classify images by style and compare them more based on how they vary in artistic interpretation.

Overall, these options could be combined to create an automated system that helps users detect where images come from and AI generated content, choose the

best model for their stylistic needs, and understand model biases and hallucinations. While my work represents just the beginning, comparing five models across a limited by diverse set of prompts, it establishes a strong foundation. The results so far demonstrate that even with a small dataset, meaningful semantic differences and habits between models can be found and visualized.

10. Deliverables:

- GitHub:
    - Link: https://github.com/RohitGeorge1/AI-Image-Generation-Comparison
    - Outputs Folder: https://github.com/RohitGeorge1/AI-Image-Generation-Comparison/tree/main/Images
    - ImageGeneration.ipynb: https://github.com/RohitGeorge1/AI-Image-Generation-Comparison/blob/main/Image%20Generation%20(1)
    - ImageComparison.ipynb: https://github.com/RohitGeorge1/AI-Image-Generation-Comparison/blob/main/ImageComparison.ipynb
- Colab:
    - Outputs Folder: https://drive.google.com/drive/folders/1Z_0BNTikY1wuh692tMy-OnUyyPzdQmFb?usp=sharing
    - ImageGeneration.ipynb: https://colab.research.google.com/drive/1oPTDEvF2a0j11SMi37iipn84lqKDHqTu?usp=sharing
    - ImageComparison.ipynb: https://colab.research.google.com/drive/1g1Ds99H_s34Ef33UOWrOCDzumrcvmBkH?usp=sharing
- Presentation Slides
    - https://docs.google.com/presentation/d/1MwDkFG9IGz39aOjbw6RQBxyor-G9D6Rp3xjIGh45Mos/edit?usp=sharing

11. Skill Learning:

During the course of this project, I acquired many meta skills which are summarized below.

- Trend/Pattern Recognition and Prediction (Level 3):
  Much of this project revolved around analyzing patterns in image generation models. As such, I learned many techniques that aided me in analyzing these patterns. For example, extracting embeddings using CLIP and performing dimensionality reduction and clustering techniques allowed me to detect patterns

present in these text-to-image models. Extracting and analyzing these patterns allow me to generalize how models behave and thus predict details regarding their outputs.

- Differentiating Facts from Fiction (Level 1):
  As mentioned earlier, the growth of AI and deepfakes presents a major danger as fake news and images are becoming much more believable. This project finds patterns present in the images produced by popular text-to-image models. As such, images of unknown origin can be broken down and have their embeddings extracted to see whether they correlate to those of model-generated images. This, combined with advanced machine learning techniques, could be used to detect fake images from real ones, deciphering fact vs fiction.

- Cyber-Physical Gap (Level 2):
  My project bridged the cyber-physical gap by analyzing how AI-generated images represent physical, imagined, or real-world scenes through text prompts. Overall, it investigates the fidelity and consistency with which popular AI models interpret human language into visual outputs. By comparing embeddings and measuring semantic alignment, I developed insights into how digital systems understand and represent the physical world.

- Reading and Writing Skills (Level 1):
  I enhanced my reading skills by parsing technical papers, blog posts, and documentation to understand previous related works and how text-to-image models work. I also improved my writing skills through writing this report, peer reviews, and project checkpoints.

- Analyzing Related Work and Recognizing Potential Opportunities (Level 1 and Level 2):
  I reviewed a wide range of prior work regarding text-to-image generation techniques, CLIP and embedding based similarity work, and model architecture summaries. I then proposed new evaluation methods like embedding similarity to text prompts and suggested future directions like classifier based source detection, developing my analysis skills.

- Time Scheduling (Level 1):
  I enhanced my time scheduling skills by managing to keep up with project deliverables while also working a full time job. This has been a major challenge and learning how to keep up with deadlines despite having limited time has greatly aided me.

- Checking Assumptions (Level 1):
  Going into this, there were many things I assumed. For one, I assumed the models would be quite different. This proved to be false. Some assumptions did turn out to be correct, however. For example, I assumed prompts would be the driving decider of the produced image. This turned out to be true. Overall, I learned how to check my assumptions and to not always take them for granted.

12. Bibliography:

[1] Chedraoui, K. (2025, June 2). Best AI Image Generators of 2025. CNET. https://www.cnet.com/tech/services-and-software/best-ai-image-generators/

[2] Stable Diffusion 2-1 - a Hugging Face Space by stabilityai. (n.d.). https://huggingface.co/spaces/stabilityai/stable-diffusion

[3] Steynberg, D. (2024, November 21). The ethical implications of AI on creative professionals. Medium. https://bytemedirk.medium.com/the-ethical-implications-of-ai-on-creative-professionals-38ec6ed983e2

[4] Kuchlous, S., Li, M., & Wang, J. G. (2024, September 15). Bias begets bias: The impact of biased embeddings on diffusion models. arXiv.org. https://arxiv.org/abs/2409.09569

[5] Tanjim, M. M., Singh, K. K., Kafle, K., Sinha, R., & Cottrell, G. W. (n.d.). Discovering and mitigating biases in CLIP-based image editing. Adobe Research; University of California, San Diego. Retrieved from https://openaccess.thecvf.com/content/WACV2024/papers/Tanjim_Discovering_and_Mitigating_Biases_in_CLIP-Based_Image_Editing_WACV_2024_paper.pdf

[6] Adewumi, T., Alkhaled, L., Gurung, N., Goya, V. B., & Pagliai, I. (2024, June 27). Fairness and Bias in multimodal AI: a survey. arXiv.org. https://arxiv.org/abs/2406.19097

[7] Düzyel, O. (2023, May 4). A Comparative Study of GAN-Generated Handwriting Images and MNIST Images using t-SNE Visualization. arXiv.org. https://arxiv.org/abs/2305.09786

[8] Xu, J., Le, H., & Samaras, D. (2024, July 21). Assessing sample quality via the latent space of generative models. arXiv.org. https://arxiv.org/abs/2407.15171

[9] Guinness, H. (2025, May 23). The 8 best AI image generators in 2025. Zapier. https://zapier.com/blog/best-ai-image-generator/

[10] Andrew. (2024, June 10). How does Stable Diffusion work? Stable Diffusion Art. https://stable-diffusion-art.com/how-stable-diffusion-work/

[11] Mishra, O. (2024, August 21). Stable diffusion explained - Onkar Mishra - medium. Medium. https://medium.com/@onkarmishra/stable-diffusion-explained-1f101284484d

[12] Amenta, A. (2023, November 15). Making sense of DALL-E 3 - Alessandro Amenta - medium. Medium.

https://medium.com/@alessandroamenta1/making-sense-of-dall-e-3s-magic-08ce61845041
[13] GeeksforGeeks. (2025, July 23). DALLE 2 Architecture. GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning/dalle-2-architecture/
[14] Tosh Marketing. (2025, May 2). How does midjourney AI work? - Global Tech Council. Global Tech Council. https://www.globaltechcouncil.org/ai/mid-journey-artificial-intelligence/
[15] Firefly vs DALL·E 3 - Adobe. (n.d.). https://www.adobe.com/products/firefly/discover/firefly-vs-dalle.html
[16] Boschman, J. (2023, July 18). CLIP paper explained easily in 3 levels of detail - one minute machine learning - medium. Medium. https://medium.com/one-minute-machine-learning/clip-paper-explained-easily-in-3-levels-of-detail-61959814ad13
[17] Rosebrock, A. (2021, April 17). Computing image “colorfulness” with OpenCV and Python - PyImageSearch. PyImageSearch. https://pyimagesearch.com/2017/06/05/computing-image-colorfulness-with-opencv-and-python/
[18] Katherine-Rittenbach. (2023, January 12). What is T-SNE? - De Novo Software. De Novo Software. https://denovosoftware.com/faq/kb-what-is-t-sne/
[19] Jaadi, Z. (2025, June 23). Principal Component Analysis (PCA): A Step-by-Step Explanation. Built In. https://builtin.com/data-science/step-step-explanation-principal-component-analysis
[20] McInnes, L., Healy, J., & Melville, J. (2018, February 9). UMAP: uniform manifold approximation and projection for dimension reduction. arXiv.org. https://arxiv.org/abs/1802.03426