Envisioning Distant Worlds: Fine-Tuning Stable-Diffusion with NASA's Exoplanet Data

by

Marissa Beaty

A thesis
presented to the University of Arts London
in fulfillment of the
thesis requirement for the degree of
Masters of Science
in
Data Science and Artificial Intelligence for the Creative Industries

London, England, United Kingdom, 2023

1 Abstract

This paper examines the ability to produce realistic images of confirmed exoplanets within NASA's Exoplanet dataset by fine-tuning Stable Diffusion with scientifically supported prompts and images. There are few ways to know for certain a planet's visual characteristics. The most common are to visually observe the planet or conduct spectroscopy. Most exoplanets within NASA's Exoplanet dataset are too far away to conduct these methods leaving scientists unable to visualize these worlds. Though these traditional methods are not available, the Exoplanet dataset includes other information that relates to stellar and planet characteristics, including stellar classification, stellar and planet mass, temperature, and orbital period. With this data, it is possible to determine a reasonable prediction of the planet's visual characteristics.

This project aims to produce an exoplanets visual prediction by developing scientific-based prompts that textually describe the numeric data for each exoplanet within the Exoplanet dataset. These prompts are generated for a dataset of images pulled from NASA's Image and Video Library of either photographs or artist renderings of near-distant worlds. This image and prompt pair then fine-tuned Stable Diffusion 2.1 and 1.5 with LoRA to optimize training capacity. In doing so, it created a generative art model that intakes a prompt composed of an exoplanet's known data and returns a predicted visual representation of that planet.

This research acts as a foundation for further study into visualizing spatial objects. The success of training a model using this type of numerical to textual conversion has potential impacts on further scientific research, game design, and potential world-building for animators or artists.

Keywords: NASA, space, exoplanets, stable diffusion, LoRA, text-to-image generation, game design, modelling

2 Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

3 Glossary

pl_name - Planet Name

hostname - Name of star or system

sy_snum - Number of stars in the system

sy_pnum - Number of planets in the system

sy_mnum - Number of moons in the system

pl_orbper - Orbital Period, the length of time (in earth days) a planet takes to make one full orbit around its star

pl_orbsmax - Orbit Semi-Major Axis, the largest radius within a planets orbit

pl_rade - Planet Radius as measured by the radius of the Earth

pl_bmasse - Planet Mass as measured by the mass of the Earth

pl_dens - Planet Density, the amount of mass per unit volume of a planet [g/cm**3]

pl_eqt - Equilibrium Temperature, temperature of a black-body planet in Kelvin

pl_imppar - Impact Parameter, the distance between the center of the stellar disk to the center of the planet disk, normalized by stellar radius
st_spectype Spectral Type or classification

st_teff - Stellar Effective Temperature, temperature of a star as a black body measured in Kelvin

st_rad - Stellar Radius as measured by radius of the sun

st_mass - Stellar Mass, as measured by mass of the Sun

Black Body - A surface that absorbs all radiant energy falling on it (i.e., a surface without an atmosphere)

Exoplanet - A planet that orbits a star outside Earth's solar system

LoRA - Low-Rank Adaptation model, a way of fine-tuning stable diffusion that requires fewer resources

Stable Diffusion - A deep-learning text-to-image model

Spectroscopy - The study of the absorption and emission of light and other radiation by matter

Contents

4 List of Supplemental Materials

Stable Diffusion 1.5 Additional Test Images

Additional Figures

5 Introduction

In 2018, NASA's Jet Propulsion Laboratory hired a team of illustrators to envision worlds in our Universe as travel destinations (JPL-Caltech, 2018). Utilizing the available data for each of these planets, the team made reasonable assumptions of the planet's characteristics and allowed science to influence the creative design of each poster. In total, the team created 14 posters featuring near and distant worlds. At first glance, the posters are something from a fantasy; you can sip cocktails on a floating diamond-shaped observatory on Venus or parasail over the lava ocean covering 55 Cancri e. Though these activities are improbable, or better said, impossible, 55 Cancri e is covered in molten lava, and Venus does have a diamond-studded sky.

Figure 1: Space Tourism "55 Cancri e" Poster

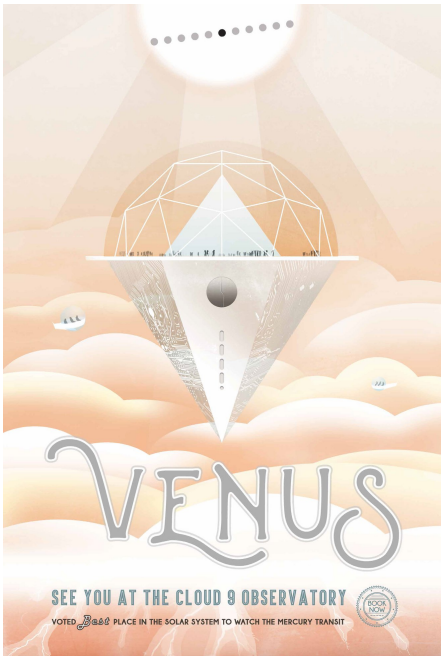

Figure 2: Space Tourism "Venus" Poster

The aim of this project mirrors that of the imagery created from the team of creatives hired by the Jet Propulsion Laboratory. It asks: Is it possible to imagine what life is like in distant worlds? Can science and mathematics allow us to predict a planet's visual characteristics for worlds too far to observe?

The answer to these questions lies in applying the same methodology as the Jet Propulsion Laboratory team but on a larger scale. The solution required developing a model that intakes a planet's scientific data and outputs an image with features representing that data. This type of textual input to generative art output naturally translated to a text-to-image generator, specifically Stable Diffusion, for its ability to produce realistic images efficiently and consistently proved most applicable. In addition, the model was made more efficient by implementing a LoRA model during training to reduce RAM dependencies and overall computing power and duration.

There are many available models and resources to assist with fine-tuning Stable Diffusion. A popular model is that by Cuenca and Paul (Cuenca & Paul, 2023). This model is built from the work within HuggingFace's "Low-Rank Adaptation of Large Language Models" (HuggingFace). This model trained Stable Diffusion 1.5 using a dataset of approximately 800 images of Pokémon characters paired with prompts generated using the Bootstrapping Language-Image Pre-training model, commonly known as BLIP (Li, 2022). BLIP creates simple text prompts by exploiting the Stable Diffusions text encoder and parameters. Cuenca and Paul suggest training for 60 to 70 epochs to maximize results. They also utilized Automatic1111 to visualize their fine-tuned model. Though this project will test a variety of different parameters, including training with Stable Diffusion 2.1 and training with fewer epochs, the most successful model was also trained with Stable Diffusion 1.5 at 70 epochs as recommended within the "Low-Rank Adaptation of Large Language Models" article.

The difficulty in executing this project is exemplified by examining the original work done by the Jet Propulsion Laboratory and understanding the necessary merger of science and art. The posters designed by the Jet Propulsion Laboratory are the first of their kind and use traditional illustration and graphic design techniques. According to JPL, the artists and scientists went back and forth when creating the final designs to ensure the creative aspects of the posters relied on a scientific foundation. Unlike the research done by Cuenca and Paul, the desired inputs for this project could not be generated using BLIP captioning. The prompts for training and testing the model would play a crucial role as a mediator between the planet's scientific data and the visualization created by the model. In doing so, it would attempt to automate the discourse between the artists and scientists developing the "Space Tourism" posters.

It isn't without noting that the difficulty truly lied within the novelty of the task. Despite extensive research, no prior scholarship exists that neatly describes a planet's visual characteristics, especially not an algorithm that could predict these characteristics. There are numerous features that contribute to what a planet and its star look like and how they function, including chemical composition, planet and stellar mass and temperature, planet spin, and so on. Though the Exoplanet dataset does not include chemical composition, as previously discussed, it does include data for these other attributes, which can be used to define a planet and a star's physical characteristics. For example, if an exoplanet within the dataset has data for either the stellar classification, stellar mass, or stellar temperature, the star's color and size can be determined by using the Harvard Stellar Classification system (Harvard). Similarly, the Exoplanet dataset contains data documenting the exoplanet's mass and radius, orbital period, temperature, and distance from its star. Mass and radius can categorize the planets into one of four categories: Terrestrial, Super-Earth, Neptune-like, and Gas-Giant (NASA, 2022). The orbital period can determine the possible cloud formations (Guzewich) and if a planet is tidal-locked (Dosopoulou, 2017) and the temperature, distance from a star, and planet category can determine possible chemical compositions that define a planet's color (Sudarsky, 2000).

As shown, with the data listed in NASA's Exoplanet dataset, a reasonable prediction can be made regarding a system's visual characteristics. How exactly this translation from numerical data to descriptive text occurs can be found within the Methods section of this report. The Methods section also discusses the various tools used to train and visualize the Stable Diffusion model.

The foundation of this work rests on the open-source notebook written by Kohya and published on Google Colab (Koyha 2023). The Kohya model proved unsuccessful in fine-tuning Stable Diffusion 2.1 but successful in fine-tuning Stable Diffusion 1.5. This success enables five iterations of training Stable Diffusion 1.5: two at 10 and 70 epochs with a maximum token length of 77 tokens, and three at 70 epochs with a maximum token length of 288. To visualize the models, the Automatic1111 WebUI notebook - made open source by The Last Ben - was used (The Last Ben, 2023). Four prompts were selected from the Exoplanet dataset - Prompts 9, 14, 488, and 687 - to test all fine-tuned models. Maintaining consistency in the test prompts allows the images and models to be compared and evaluated. The model's were also tested with random prompts to confirm consistencies or faults in the models and to select the most reliable model from those trained. The full results and interpretations are available in the Results and Discussion section of this report.

Success in this work is defined by the ability to reproduce the work of JPL: that is, in its ability to envision distant worlds based on their scientific and mathematical properties. Though the project could benefit from additional research and a larger dataset, as discussed within the Recommendations section, it proves the ability to generate probable images of these distant worlds using the data available. This project creates a foundation for further interest in a more scientific development of world-building or visualizations, whether that pursuit is through scientific research, game design, or the visual arts.

6 Research Question

Briefly mentioned in the Introduction, this project seeks to build on and expand the work from the 2018 "Space Tourism" posters produced by a team of creative artists in collaboration with NASA's Jet Propulsion Laboratory. In doing so, it asks: Is it possible to generate realistic images of distant worlds by fine-tuning Stable Diffusion with scientifically defined prompts?

Delving into this question required splitting the research into three distinct parts. Part, one includes obtaining the datasets for training and testing and translating the numeric data within the datasets to descriptive prompts. Part two takes the image and prompt pairs as inputs to fine-tune Stable Diffusion and outputs a new model trained within the scientific constraints and image quality defined by the input data. Part two also visualizes the results of each model with the control prompts. In part three, the images produced by each model are evaluated based on their realism and accuracy depicting the input prompt. Subsequent random testing is also conducted within this section to confirm or refute conclusions made about a model's success.

Part one and part two are likely to be the most time-intensive aspects of the project, as well as the areas with the most potential for complications. For part one, effectively defining textual prompts from the numeric data will be the most laborious aspect. This part will rely heavily on conducting research and acquiring a deep understanding of the astrophysical and planetary geological aspects of planetary and stellar objects.

For part two, the difficulty lies in obtaining a successful fine-tuned Stable Diffusion model. Given the multitude of different parameters, Stable Diffusion versions, and computing engine requirements, there are several places in which errors could arise, as well as several

adjustments that could cause different results. To combat the difficulties identified in part two, relying on the resources that are available documenting various solutions is necessary. Testing a variety of parameters, however, requires adjusting and retraining the model. This is the challenge of this section. Careful and cautious consideration is needed when selecting parameters, as training the model is time-consuming and costly. Within this section, the model's are also visualized using the control prompts (Prompts 9, 14, 488, and 687).

Part three is when each model is evaluated and retested with random prompts to confirm the model's evaluation. There are two criteria for a successful model: how realistic the image generated is, and how well the image adheres to the input prompt.

The difficulties predicted above are hypotheses of what might arise throughout this research. A review of all challenges and results encountered throughout this project can be found in the Methods and Results section of this report. It is predicted this project will successfully produce a model that generates realistic images of distant worlds based on the input prompts. It should be noted that the goal of this project, nor the success of this project, rests on the complete accuracy of the predicted images. Given the inability to confirm the exact compositions of these planets, the results are expected to be accurate to their prompts and predictions of their real-world counterparts. That is not to say the model is not successful, but rather that the data does not exist for it to be exact. The success of this project, instead, will lie in its ability to answer the research question listed above: Is it possible to create a scientifically backed Stable-Diffusion model that can create realistic interpretations of a distant exoplanet? The answer lies within the Discussion section of this report.

7 Methods

As mentioned in the Research Question section of this essay, the research methods for this project have been split into three parts. Part One, involves preparing the datasets and accumulating the necessary research to define and obtain input prompts. Part two, defines the process and parameters used to train and visualize the models. Finally, in part three, the models are evaluated and undergo subsequent testing to determine the most successful model and the best parameters for future iterations of this work. This following section of the report will discuss each part in turn:

7.1 PREPARING THE DATASETS

The first requirement for this project was acquiring the datasets. Gaining open-source access to NASA's API and datasets required a simple application process to NASA's API system to obtain a personalized API key (NASA, 2023). This project utilized two datasets built from this API: NASA's Confirmed Exoplanet dataset and NASA's Image and Video Library. The Exoplanet dataset is easily accessible via the Web API and can be downloaded directly to a local server as a .csv file (NASA Exoplanet Archive, 2015). This dataset contains essential information about an exoplanet, including stellar and planet mass, stellar category, effective temperatures, and more. This data is vital to developing a scientifically based prompt generator, as defined in the project goals.

The Image and Video Library houses all videos, drawings, graphs, and images published or sponsored by NASA. Accessing images within this library requires the API key. With this key, the entire library can be queried to obtain the desired images for training. These images will be paired with accompanying scientific data, which will be used to generate a unique prompt for that image. The prompt and image pair will be the input for the training module. As mentioned, the goal of this project is to generate realistic visualizations of a given exoplanet. Due to this, the images selected for the training dataset were limited to photographs, scientific drawings or visualizations of planets and exoplanets. The queries into this Library were also specifically chosen to adhere to the project and were limited to the following searching: "exoplanet," "planet artist concept," and "planet photographs."

As the Image and Video Library houses tens of thousands of images, code was developed to query the library and visualize the results of the query. From these visualizations, the desired images were selected, and the image data was saved into a Pandas DataFrame (Figure 41). This process was repeated for all three search queries. As mentioned, each image needed to be paired with scientific data so that a prompt could be generated. As the images did not include this data, it would have to be added manually. To do this first required identifying the necessary pieces of information that could be influential in generating prompts. These included:

- pl_name
- host_name
- sy_snum
- sy_pnum
- sy_mnum
- pl_orbper
- pl_rade
- pl_bmasse
- pl_dens
- pl_eqt
- pl_imppar
- pl_orbsmax
- st_spectype
- st_teff
- st_rad
- st_mass

Next, the image descriptions and captions were scanned to determine any vital information, such as planet name or hostname. With this information, the necessary scientific data could be acquired by either matching the planet name or hostname to the name and data within the Exoplanet dataset or by searching NASA's data archives for information on the planet. Through this process, the available scientific data was acquired for each planet and star, completing the training dataset's pre-processing.

Once both datasets were complete, research commenced to determine textual descriptions for each relevant data point. For example, the Exoplanet dataset has a column called sp_spectype, which lists the stellar category for that planet's star. This classification system - based on the Harvard Classification scheme - ranks stars as O, B, A, F, G, K, or M (with a few other rankings, such as WD for white dwarfs and SD for subdwarfs) (Harvard). Within this system, an O-class star is the biggest and brightest, while an M-class star is the smallest and dimmest. These classifications determine other characteristics as well. If a star is within class G, such as the Sun, it will likely be yellow, with a mass between 0.8 and 1.04 solar masses and a temperature between 5,200 and 6,000 Kelvin (Harvard). This process was repeated for all stellar classifications within the two datasets by defining a simple function that scanned the sp_spectype column and picked out the first letter within the spectral classification column. Based on the value, it assigned a stellar color for the system in a new column titled "stellar_color". If the exoplanet did not have a sp_spectype listed, the process

would instead look at st_mass. If neither value existed, it reverted to st_teff. Finally, for any remaining stellar object, it listed "unknown stellar color" (Figure 42).

With stellar color determined, the last portion for the stellar description was size. Several options could determine stellar size. Since this project wants to test different versions of prompts, three stellar sizes were pursued. The first uses st_mass, as listed in the dataset, to define stellar size numerically as a ratio to the Sun. The second defines stellar size through a text classification function. This classification function uses the stellar color defined previously to determine size, where the big, blue, O-type stars are "massive" and the small, orange-red, M-type stars are "very small" (Figure 43). This also accounts for smaller white dwarf stars and subdwarf stars, which are labeled "tiny," and stars with an "unknown stellar color" labeled "unknown size." The third definition determines stellar size as a numeric ratio to its planet size (Figure 50). Besides st_mass, which is already within our dataset, each of these size definitions was saved into a new DataFrame column labeled "stellar_mass_description" and "stellar_planet_ratio," respectively.

This concluded the text required for the stellar description portion of the prompt. A far more complex task, working on the planet descriptions, is next. What determines the visual characteristics of a planet can be any number of factors such as size, temperature, atmosphere, chemical composition, etc. The only way to confirm these visual characteristics is to directly observe them with spectral imaging or capture the planet's spectroscopy (NASA, 2022).

As mentioned, the exoplanets within the training dataset and exoplanet dataset are too far away from modern technology's capabilities for capturing this data. Due to this, the exoplanet descriptions developed throughout this research project are viable predictions of the planet's category, color, weather patterns, and potential for tidal locking. Each characteristic will be defined in turn, beginning with the planet classifications.

There are four planet classifications within our known Universe (NASA, 2022). The first category is the Terrestrial planets, such as Mercury, Venus, Earth, and Mars. These planets are distinct for their rocky, metal cores and rocky landscapes recognized by mountainous regions and caverns. These planets are likely to live in the Goldilocks zone, which defines the range of distances between a planet and its star where liquid water (and, thus, life) could exist (NASA, 2022). The second category is the Super-Earth planets. Planets within this category have rocky surfaces much in the same way as the Terrestrial planets but are between 2 and 10 times more massive than Earth. The third category is the Neptune-like planets. These planets are similar in size to Neptune and Uranus and are likely composed of frozen gases surrounding a rocky core. The fourth and final category is the gas giants, like Saturn and Jupiter. These planets are distinct for their enormous size and are composed of gas-filled clouds.

Categorizing the exoplanets requires the pl_bmasse or pl_rade. A simple function used these values to categorize the planets and list their category as a new column titled "planet_category" (Figure 45). With the planet category defined, the next step was to define what the planet would look like. This involved interpreting the planet's color, wind patterns, and potential to be tidal locked. This project began with defining planetary color. As mentioned, there can be any number of factors that can affect a planet's emitted color. Chemical composition (as collected through spectroscopy) is one way of doing this. Though spectroscopy is not available for these exoplanets, many have pl_eqt, also known as the planet's black

body temperature. As chemicals form solids and gases and break down at different temperatures, the planet's black body temperature could give a rough estimate of a possible chemical composition, and thus, a possible planet color. With this information, research commenced determining the most common chemicals within the Universe, the temperatures at which they change states of matter, and their absorption and emission of visual spectra (Figure 44) (NIST, 2023). For example, Methane emits blue wavelengths and absorbs red wavelengths. It also has a greater intensity in color at cooler temperatures (Duan, 2019). Neptune and Uranus both have high concentrations of methane, which gives them their distinct blue color, but more importantly have black body temperatures of 72 and 77, respectively (NASA, 2016). These temperatures happen to be when Methane is a solid: its coolest state. Through the relationship between temperature and chemical properties, a very basic (albeit not perfect) estimation can be made of a planet's visual characteristics. Based on the data within Figure 44, a column named "planet_color" was created describing an exoplanet's color based on the planet_category and then the pl_eqt if that data was available (Figure 47). If an exoplanet did not have a pl_eqt listed, planet_color defaulted to pl_bmasse and made assumptions based on the exoplanet's size in relationship to the size of planets in our solar system. Though this is not a perfect relationship, it is known that planets within a similar category, and thus, a similar mass range, behave and look similarly (NASA, 2022). Without the additional data to support a more accurate estimation of planet_color, this theory allows for a loose estimate of a planet_color for these distant exoplanets. As for planets without any of the above data, their planet_color was listed as an "unknown planet color".

With planet color determined, planet size is the next characteristic. Much like stellar size, planet size was broken into three options. The first was using pl_bmasse, which describes the planet's mass as a ratio to Earth's mass. Much like st_mass, this column already exists within the dataset. The second defined planet size with text using the same categorical system defined for the stars but shifted to fit the masses of the planets in our solar system. Exoplanets the size of Mercury were labeled "tiny" and exoplanets the size of Jupiter were labeled "enormous". The values for this column were saved as "planet_mass_description" (Figure 46). The third is the same ratio between planet and stellar mass created in Figure 50 and discussed above. These three options will be paired with the three equivalent stellar options when defining different prompts to train and test our model.

Finally, work can begin on planet spin. The Exoplanet dataset holds enough information to estimate the exoplanet's spin and its possible weather patterns. The faster a planet spins, the more turbulent its weather and the more distinct the cloud patterns are across the pattern's surface (Guzewich). This prediction would be most accurate using the planet's axial spin, which is not within the Exoplanet dataset. Instead, the planet's pl_orbper, paired with the planet's category, provides a close alternative. For example, a planet within the Terrestrial category with a pl_orbper shorter than Mercury is spinning very fast around its star and likely to have few clouds, whereas a Gas-Giant with a pl_orbper shorter than Jupiter would likely have thick stripes of clouds covering the exoplanet's surface, despite also spinning relatively fast for its size. With this data, a "planet_spin" category was created describing the possible cloud formations found on that exoplanet given this same relationship between planet_category and pl_orbper. If a planet did not have a pl_orbper value, pl_bmasse provided a backup estimation. If neither value was available within the dataset, the planet_spin column was given the value "unknown planet spin" (Figure 48).

The final description was to determine whether the exoplanet was tidal-locked. Tidal locking is when a planet's axial spin is the same as the planet's orbital period (Dosopoulou, 2017). In a tidal-locked planet, only one side of the planet faces its sun, causing one side to be extremely hot and bright while the other side remains extremely cold and dark. As mentioned, the Exoplanet dataset does not have axial spin nor the necessary data to calculate it. There is, however, another way to determine tidal locking using the Roche limit. If a planet crosses the Roche limit, i.e., when a planet gets to a certain distance from its star, the planet is most likely tidally locked (Dosopoulou, 2017). The Roche limit can be calculated through a mathematical formula using pl_bmasse and st_mass (Vanderbilt University). To determine if the exoplanets within our dataset are tidally locked required first applying the mathematical formula to determine the planet's Roche limit, saving that value within a new column titled "roche_limit" and then comparing the newfound value within the "roche_limit" column with the data within the "pl_imppar" column. If the determined Roche limit is less than the pl_imppar for that exoplanet, then the exoplanet is likely tidal-locked (Figure 49). A new column is created called "tidal_locked" that describes whether the exoplanet is not tidal-locked and "spins around its orbit so both sides get heat from the sun" or is tidal-locked with "only one side of the planet facing the sun. The side facing the sun is extremely hot, and the side that faces away from the sun is dark and cold".

Before developing the prompts, one last feature needed to be defined: shorter descriptions of the planet_spin and planet_color columns so that a model could train within the given 77 token requirement of the basic Stable Diffusion text encoder. Stable Diffusion pads the input prompt with a front and back token, so the available token size is actually 75. The features described above are set to train on a longer maximum token length of 288 tokens. Thus, the longest descriptors must be trimmed to fit the shorter token length. To create the shorter prompts required copying over the data from planet_spin and planet_color, editing the prompts to be less descriptive while maintaining the most necessary parts (such as removing why a planet is a specific color and just listing the color), and saving them to a new column as planet_spin_short and planet_color_short, respectively. An example of this is as follows:

> **planet_spin original description**: 'has stripes of thick clouds of various coloring defined by softened edges'
> **planet_spin shorter description**: 'has soft-edged stripes of thick clouds'

> **planet_color original description**: 'consists mostly of helium and hydrogen which are dominantly white, but it mixes with frozen methane characterized by a light blue color'
> **planet_color shorter description**: 'is mostly white mixed with light blue in color'

With this defined, the individual elements can be pieced together into a prompt. Creating the prompts was a simple task that required developing four different lines of code that pieced together the descriptions created above into a coherent paragraph. A function created a new column within the dataset for each of the four prompts for every exoplanet listed. The first prompt used pl_bmasse as a numerical description of planet size and st_mass as a numerical description of stellar size. This prompt was labeled "mass_prompt." The second prompt,

titled “ratio_prompt,” utilizes the numeric ratio of the planet to stellar size to describe the two objects. The third prompt, “size_text_prompt,” uses the “stellar_mass_description” and “planet_mass_description” columns to textually describe the size of the star and the size of the planet. The fourth prompt, labeled “75_tokens” utilized the “planet_spin_short” and planet_color_short” to remain below the 77 token requirements of the original Stable Diffusion text encoder (Figure 51).

By running the training and exoplanet datasets through each function, every exoplanet had four prompts ready to train the model.

7.2 TRAINING A MODEL and VISUALIZING RESULTS

The Kohya-LoRA Fine-Tuning colab notebook (Kohya, 2023) set-up to train Stable Diffusion 2.1 with a Stable Diffusion LoRA model was utilized for training. To standardize training, the following parameters were standardized across all training iterations:

Batch size = 8
Training Batch Size = 6
Mixed Precision = fp16
Data Loader Number of Workers = 2
Resolution = 512 x 512
Optimizer = AdamW8bit
Automatic Save = every 1 epoch to a. safetensor file

All images within the training dataset needed to be downloaded to the local machine with the correct label and resolution required for the training notebook. This was a simple process of opening the image path, resizing the image, and saving the resized image to the local machine with a new path name. To review this, please refer to the Github Repository for this project listed within the References of this report. Though a part of the Kohya notebook resized and pre-processed the images, it required them to be uploaded as .jpgs to Colab. Since the images in the training dataset existed as web URLs at that point, it was simpler to resize the images at the same time they were downloaded.

This notebook also utilized BLIP captioning and a Waifu Diffusion 1.4 Tagger to create image captions and tags for training. Running these creates a metadata file holding a dictionary of dictionaries where each key is the image path, and each value is a dictionary holding the image tags and image captions. The format is as follows:

{
image_path:{“tags”: keywords associated with the image, “text”: image caption},
image_path: {“tags”: keywords associated with the image, “text”: image caption},
image_path: ...
}

Since this project wants to train on pre-generated captions, BLIP captioning, and the Waifu Tagger were replaced with a generated metadata file matching the format and title of

"metadata.csv" created by the two within the Kohya notebook (Figure 52). The downloaded images and metadata file were uploaded to Colab within the LoRA folder created by the notebook's "Install Dependencies" section. To continue using the notebook, a cell updating the version of pip requests to version 2.31.0 was added and run. To begin training, the notebook was run using the pre-set Stable Diffusion 2.1 base and AnyLoRA models. In addition to the image pre-processing, BLIP, and Waifu sections, uploading a custom model and downloading an available VAE were not run, as they provided no benefit to training in this circumstance. All training parameters that were adjusted are available in Figure 40. Anything not included has a value either of zero, no, or was not run.

The first iteration of training with Stable Diffusion 2.1 was with 10 epochs and our smaller prompt "75_tokens" to fit the 77 token limit pre-set for Stable Diffusion, including the front and back padded tokens. Training took approximately thirty minutes using Colabs NVIDIA A100 GPU. When no errors were raised during training, the above configurations were also run through 40 training epochs on the NVIDIA A100 GPU, which took approximately two and a half hours. These models were saved with different names, to be able to test each individually.

Both models were then visualized using an adapted Colab notebook made open source by The Last Ben (The Last Ben, 2023). This notebook utilizes Automatic1111 WebUI and a personal Ngrok key to launch a trained Stable Diffusion model that can be tested with a given prompt. In order to visualize the trained models they had to be uploaded onto Google Drive within the content/MyDrive/sd/stable-diffusion-webui/models/Stable-diffusion/models folder. The sd folder is uploaded to Google Drive during the "Requirements" section of The Last Ben notebook. Unlike the Kohya notebook, all cells in The Last Ben notebook were run. When running the notebook, the Model Version was set to V2.1, and LoRA and ControlNet were set to default on the standard Stable Diffusion packages. When the WebUI link became available, the first model selected to test was that trained with 10 epochs. The control prompts were used to test the model. These can be viewed in Figure 38 under the model name "75_tokens."

Each prompt was tested individually with the model, however, each, in turn, was unsuccessful (Figure 6). After doing some research, it was theorized better results could be achieved by removing NSFW limitations or removing half values in the COMMAND_ARGS line. Neither of these solutions, however, altered the image results. It was then thought, there was an error with 10 epochs not having enough training time to produce successful results, however, when testing the same prompts on the model trained with 40 epochs, the same image results were yielded (Figure 8). To ensure this issue was a fault of the model, the same four prompts tested an untrained version of Stable Diffusion 2.1 (Figures 4-7). This model provided successful results, alluding to an issue training Stable Diffusion 2.1. Additional research suggested training Stable Diffusion 1.5 over Stable Diffusion 2.1 for best results (O'Connor, 2023).

Thus, the model was set to retrain using Stable Diffusion 1.5. Most of the training parameters set for Stable Diffusion 2.1 were kept to maintain consistency with only the following being altered:

The first iteration of training was set to 10 epochs to test whether adjusting the version of Stable Diffusion would provide better results. This training took approximately thirty minutes using the NVIDIA A100 GPU. This model was then saved within the content/My-

Figure 3: Altered Training Parameters

2.1 Download Available Model

- model_name = AnyLoRA
- v2_model_name = Stable-Diffusion-1-5

4.4 Bucketing and Latents Caching

- model_dir:/content/pretrained_model/Stable-Diffusion-1-5.safetensors

5.1 Model Config

- project_name: exoplanet_data_sd_1.5
- pretrained_model_name_or_path:/content/pretrained_model/ Stable-Diffusion-1-5.safetensors

Drive/sd. . . folder and visualized using Automatic1111 WebUI. The same four prompts tested the model and successfully produced, albeit imperfect, but promising images (Figures 13-16). Based on the recommendations within HuggingFace's LoRA training guideline, the model was retrained with 70 epochs. This took approximately 4 hours and 11 minutes using the same NVIDIA A100 GPU. The results of this training also proved successful and were visualized using the Automatic1111 notebook with the same configurations (Figures 17-20). In addition, an untrained version of Stable Diffusion 1.5 was tested with the four prompts to compare the results against the trained model (Figures 9-12).

The above training utilized the training prompt and test prompts defined for the 77-token standard for Stable Diffusion. To test the longer prompts required retraining the model three more times with a longer token length of 288 tokens. The first two retraining sessions would visualize the longer descriptions that utilize numeric values for planet size: mass_prompt and ratio_prompt. The third and final training would visualize the size_text_prompt. Retraining three additional times required creating new metadata files with the correct prompts and uploading that metadata file when running the training module. Additionally, the max_token_length in 5.4 was changed to 288 instead of 77. The rest of the training configuration remained the same as the model that trained Stable Diffusion 1.5 with the 75_token prompt. Each of the three iterations trained with 70 epochs and took approximately 4 hours with the NVIDIA A100. The models were saved and visualized using the Automatic1111 WebUI. The same prompts - 9, 14, 488, and 687 - were used for visualizing the three models. The results from the mass_prompt training is available in Figures 21-24. The results from testing the ratio_prompts are available in Figures 25-27. The size_text_prompt results is available in Figures 29-32.

7.3 ADDITIONAL TESTING

With all models trained, it was time to determine the best model for future use and development of this research. The first step to determine this was to evaluate the results from prior training. This consisted of a visual evaluation comparing the image results to the input

images and the input prompts. A successful model would not only maintain stylistic similarities to the input images but would generate an image as closely to the described prompt as possible.

Following this evaluation, each model was tested with an additional one or two random prompts to support or alter the visual evaluations made. The images generated from these additional tests were analyzed as well with a singular model selected as most consistent to the desired image style and input prompts.

8 Results

This project attempted to retrain Stable Diffusion seven times. For ease of reviewing and discussing the results of each training, they will be discussed in turn, beginning with the first attempts at training Stable Diffusion 2.1, moving to training Stable Diffusion 1.5 with 77 tokens, and concluding with training Stable Diffusion 1.5 with 288 tokens.

8.1 STABLE DIFFUSION 2.1 – 10 Epochs and 40 Epochs with 77 tokens

The first model trained was Stable Diffusion 2.1. To provide a valuable comparison with the trained model's capabilities, the prompts were first tested on an untrained Stable Diffusion 2.1 model as shown below.

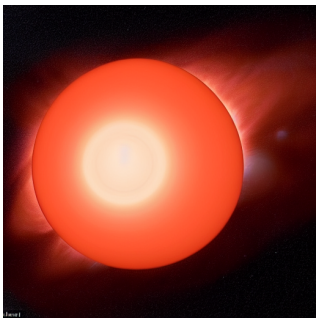

Figure 4: Prompt 9

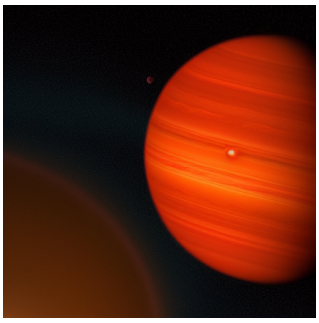

Figure 5: Prompt 14

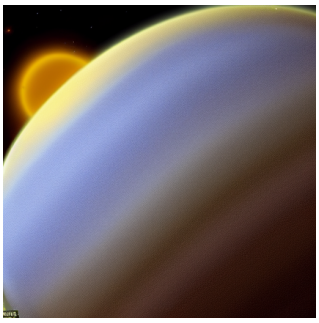

Figure 6: Prompt 488

Figure 7: Prompt 687

Untrained Stable Diffusion 2.1 Images

At first glance, these images represent the same type of space images found within the training dataset, however, they lack any adherence to the input prompts. Figure 5, for example, is supposed to represent a blue world with traces of white, but instead imagines a red world with thick clouds. Figure 4 gets the correct planet color but fails to represent the cloud formation striping the planet and has an almost cartoon-like quality. It is clear the untrained model is not perfect, but it is a solid basis for comparison. Stable Diffusion 2.1 was originally trained with 10 epochs to test the model's training rate, and 40 epochs to obtain a better sense of the model's trained abilities.

Though no errors were thrown during either training, both models failed to produce any successful results. When visualizing the results of these models, all tests produced the same, black image, representing the empty tensor produced during training (Figure 8). It was

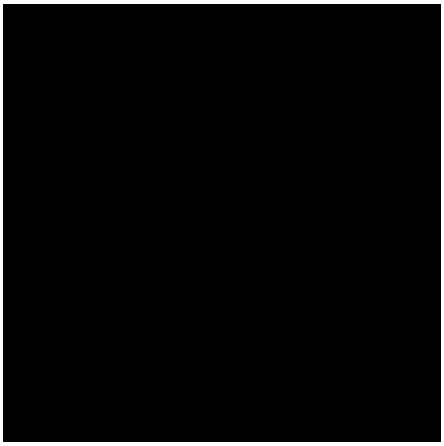

Figure 8: Failed Trained 2.1 Model

initially predicted this was an issue with NSFW restrictions, however, removing NSFW from the COMMAND_ARGS line did not resolve the error. It was then thought to be an issue with –xformers and the production of half tensors, which should have been resolved by removing –xformers and adding a –no-half argument into the COMMAND_ARGS (Automatic, 2022). This, yet again, did not resolve the error. The most likely cause of this model's failure lies in the model's embedding structure, however, a solution for this could not be found (Automatic, 2022).

8.2 STABLE DIFFUSION 1.5 – 10 Epochs and 70 Epochs with 77 tokens

Additional research showed Stable Diffusion 1.5 had a far higher success rate training than version 2.1 (O'Connor, 2023). The next series of tests retained most of the prior configurations set for Stable Diffusion 2.1, but trained on Stable Diffusion 1.5. To begin, an untrained Stable Diffusion 1.5 model was tested with the four given prompts. The results are available below.

Figure 9: Prompt 9

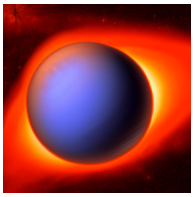

Figure 10: Prompt 14

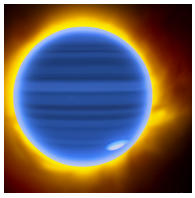

Figure 11: Prompt 488

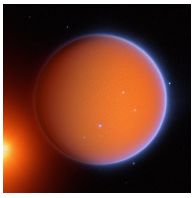

Figure 12: Prompt 687

Untrained Stable Diffusion 1.5 Model

These images were more successful in adhering to the prompts than the untrained Stable Diffusion 2.1 but still retained a sense of fiction not desired for this project. For example, Figure 10 was depicted correctly as a blue planet (unlike Figure 5) but lacked the cloud formations described in the prompt. Regardless, it appeared to be a better starting point than Stable Diffusion 2.1 and was able to be successfully trained. The first training, at just 10 epochs, was to determine if Stable Diffusion 1.5 could, in fact, be trained. When the model completed training, the selected prompts were tested with results available in the figures below.

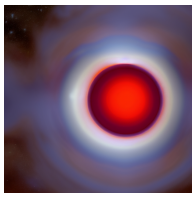

Figure 13: Prompt 9

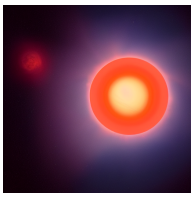

Figure 14: Prompt 14

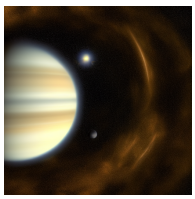

Figure 15: Prompt 488

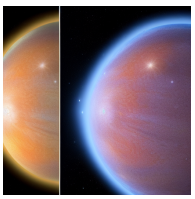

Figure 16: Prompt 687

Stable Diffusion 1.5 75_tokens Trained at 10 Epochs

These images showed promise this method would work. To see the full limitations of the training model, training was set at 70 epochs based on the recommendations from Cuenca and Paul (Cuenca, 2023).

Figure 17: Prompt 9

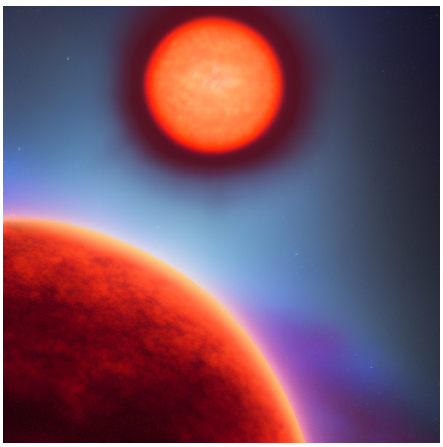

Figure 18: Prompt 14

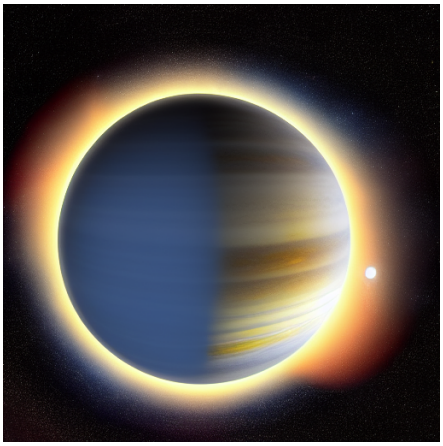

Figure 19: Prompt 488

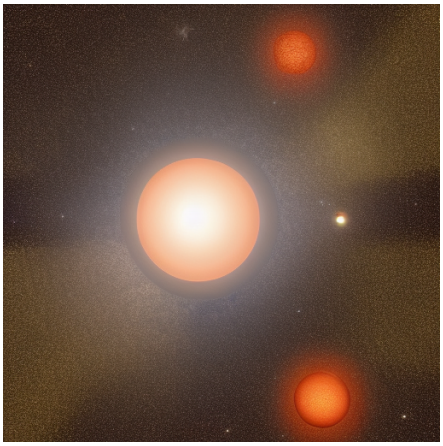

Figure 20: Prompt 687

Stable Diffusion 1.5 75_tokens Trained at 70 Epochs

The results produced from the 70 epochs were far more promising and showed a closer likeness to the images pulled from the Image and Video Library (Figures 17-20). Especially when placed against the training images from 10 epochs, the model has significantly improved its understanding of planetary aspects such as spin and cloud formations. Figure 19 significantly improved its expression of the striped cloud formations and the white coloring as described in the prompt. Figure 18, however, lost the blue color described and represented in the untrained model. It is clear the model is not perfect, but very promising.

8.3 STABLE DIFFUSION 1.5 – 70 Epochs with 288 Tokens

The next three iterations of training increased the maximum token size from 77 to 288 to allow for more detailed prompts. In total, three longer prompts were trained and tested.

The first is the mass_prompt, which calculates stellar and planet mass numerically, like the 75_token prompt, but with more descriptive detail about the planet and stellar coloring and size. ratio_prompt is the second prompt trained. This prompt maintains the same descriptions as the mass_prompt, but to describe the size of the planet and star, it uses a numeric ratio. The third prompt trained is the size_text_prompt. This prompt, again, uses the same descriptions as the other two prompts but uses a textual descriptor to describe stellar and planet size. The prompts are available to review in Figure 38.

Each new training required a new metadata file paired with the training images. The models were trained individually for 70 epochs, which took an average of 4 hours and 23 minutes for each training session. These trained models were saved and visualized using The Last Ben's Automatic1111 WebUI.

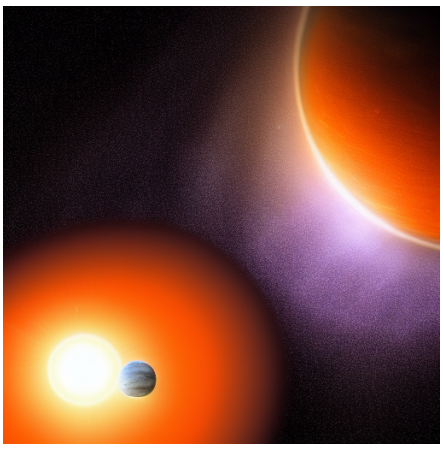

Figure 21: Prompt 9

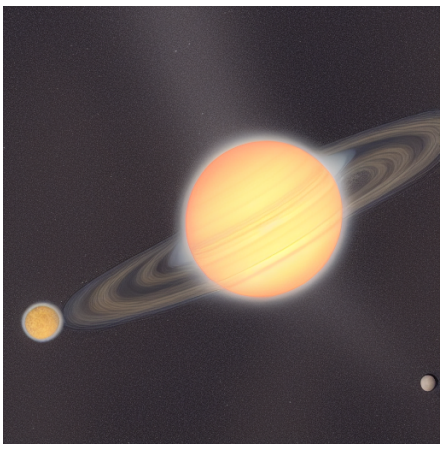

Figure 22: Prompt 14

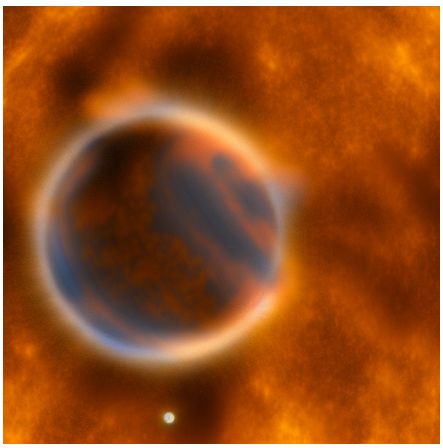

Figure 23: Prompt 488

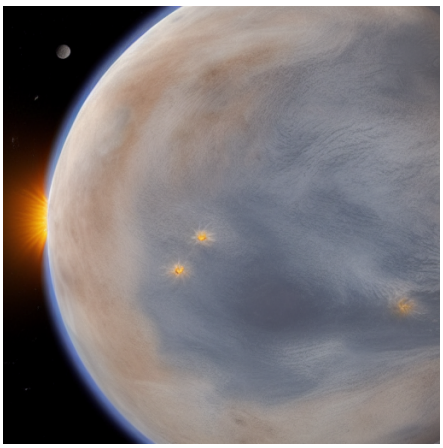

Figure 24: Prompt 687

Stable Diffusion 1.5 mass_prompt Images

The visualizations of mass_prompt, shown in Figures 21-24, are also promising but not as accurate as desired. Figure 24, for example, understood the planet was white and blue, and its star was orange, but the other images produced from prompts 9, 14, and 488 fail to represent the nuances of the descriptions, such as the cloud formations in Figure 21, or the

white clouds in Figure 23.

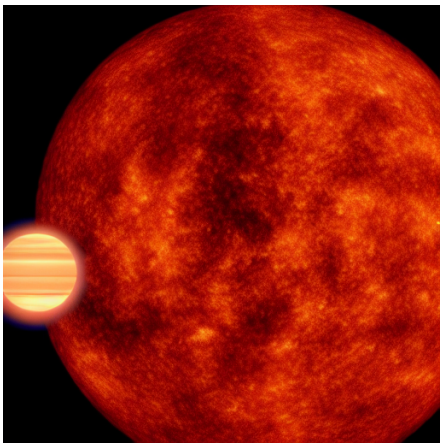

Figure 25: Prompt 9

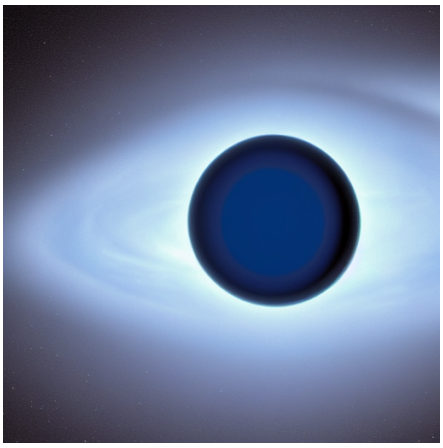

Figure 26: Prompt 14

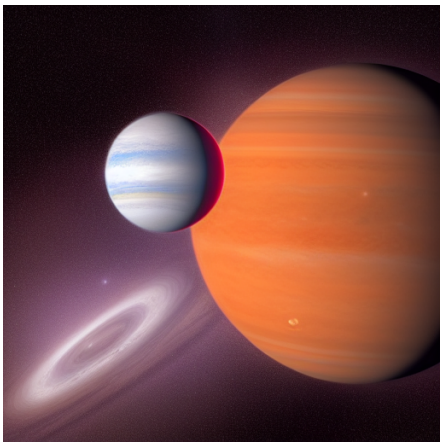

Figure 27: Prompt 488

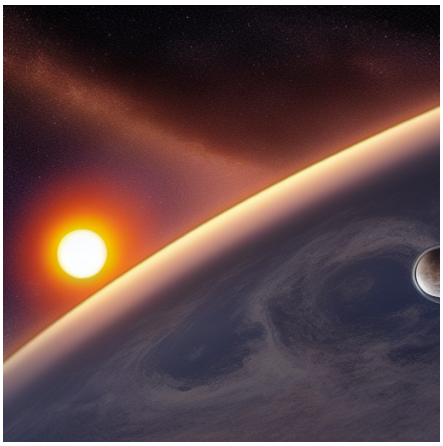

Figure 28: Prompt 687

Stable Diffusion 1.5 ratio_prompt Images

ratio_prompt's images proved even less accurate (Figures 25-28). Within these images, the model fails to differentiate between what is a planet and what is a star, as is the case in Figure 27 where both planet and star have cloud formations, and Figure 25 where only the star shows the characteristics describing the planet.

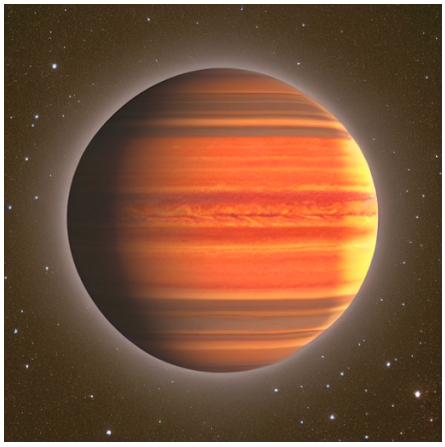

Figure 29: Prompt 9

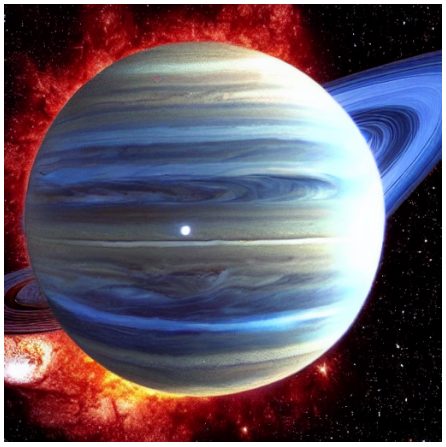

Figure 30: Prompt 14

Figure 31: Prompt 488

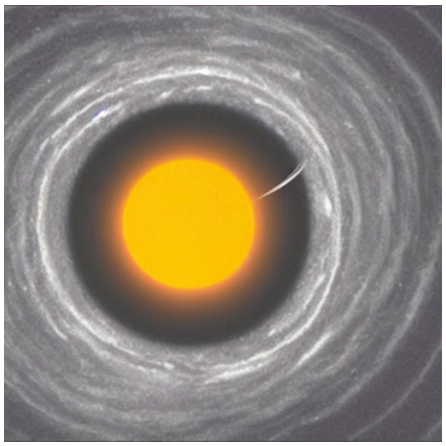

Figure 32: Prompt 687

Stable Diffusion 1.5 size_text_prompt Images

size_text_prompt, had slightly better results than ratio_prompt (Figures 29-32). Figures 29-31 all represented the correct planet cloud formations and coloring described in the prompts, but still had slight inaccuracies and inconsistencies, such as the stellar representation either not existing like Figure 29, or lacking realism, such as the sun depicted in Figure 31. These inconsistencies is exaggerated in Figure 32, which failed on all accounts to realistically represent the given prompt.

8.4 ADDITIONAL TESTING

Due to the continued inconsistencies across all images, size_text_prompt was eliminated as a contender for the most reliable and consistent model. This is validated by an additional prompt test, which reaffirms the model's inability to produce realistic images consistently (Figure 34). Between the last threes fully trained models, additional testing disproves ratio_prompt as the best model, again, for a lack of consistency particularly in its ability to produce realistic images as emphasized by Figure 33's cartoon-like background.

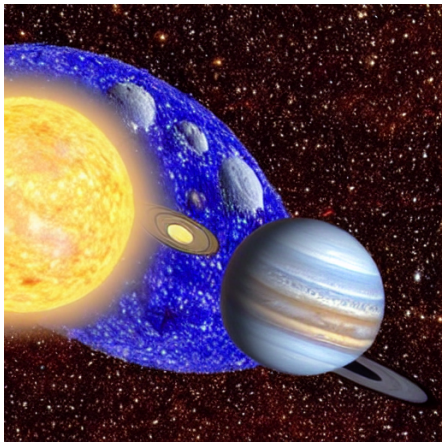

Figure 33: ratio_prompt Prompt 3219

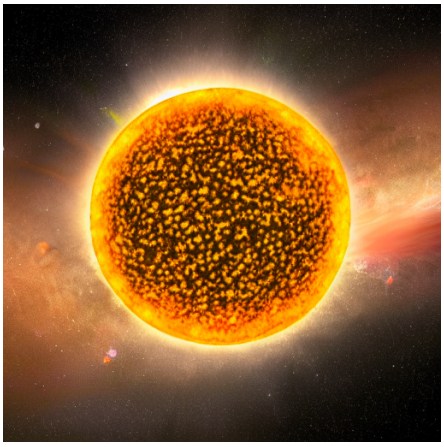

Figure 34: size_text_prompt Prompt 4594

Though adhering to the prompt's is an important factor for the project, it is equally as important the images produced are consistent with the realistic style of the input images, as their realism helps support the scientific validity of the image.

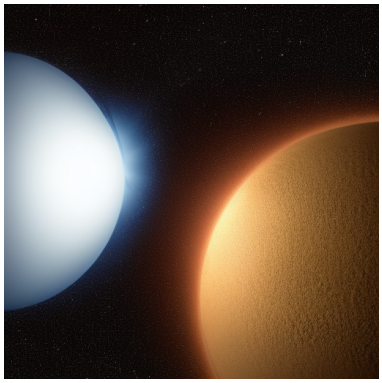

Figure 35: 75_tokens Prompt 1227

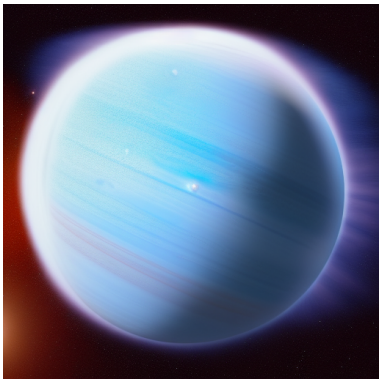

Figure 36: 75_tokens Prompt 1066

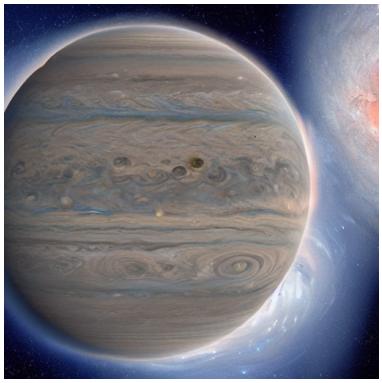

Figure 37: mass_prompt Prompt 1168

With that in mind, only two models remain to be further evaluated: 75_tokens and mass_prompt. Between the two, the model trained with only 75_tokens proved more consistent in producing the image prompt and maintaining realism in the generated image (Figures 35 and 36). Though the final image tested for the mass_prompt also fit these criteria (Figure 37), when considering all prompts and images generated from each model, mass_prompt lagged a bit on consistency against the 75_token model.

9 Discussion

The resulting images have made clear two things: one, that developing a generative model able to create images based on scientific facts is possible and useful for imagining distant

worlds incapable of being seen, and two, it has made visible the limitations that exist when training a Stable Diffusion model with this type of scientific and descriptive prompt system.

To begin with the image results and their successes, as discussed within the Results section, the most successful model is Stable Diffusion 1.5, trained at 70 epochs with a maximum token length of 77 tokens. Why did this model produce better results than the other three when it was less descriptive?

The predicted reason lies within Stable Diffusion's core structure. Though it is possible to fine-tune Stable Diffusion with longer prompts, as was successfully proven by the mass_prompt, ratio_prompt, and size_text_prompt models, it is not the most compatible method with the Stable Diffusion text encoding structure. Stable Diffusion's original maximum token length is 77 tokens (Stable Diffusion). This means that the token length that produced the Stable Diffusion model fine-tuned in this project was 77 tokens. Any additional text above the 77 token limit is encoded separately and added together to create one encoding. This increases the potential for lost information, such as noun ownership of adjectives or verbs. Even though fine-tuning this model beyond this maximum token length is possible, the research here shows that doing so decreases the model's understanding of the text, specifically its association of adjectives and verbs to their paired nouns.

An example of this is how ratio_prompt confused the descriptions between the planet and star and associated planet descriptors to the star and vice versa. It appears the additional description proved harmful rather than helpful when fine-tuning the model, as the supplementary detail muddled the algorithm's ability to distinguish each noun and assign it the correct corresponding adjectives. This could be why the 75_prompt model worked best. The image results also support the idea that simplifying the prompts further might prove even more effective and increase the consistency of the model adhering to the input prompt.

Such a simplified prompt could look like:

> "a large, blue planet with thick white clouds circling a small, red star"
> rather than
> "A light orange, small star with a giant, gas-giant planet. The planet is mostly blue with traces of white coloring, has clear, sharp-edge stripes of thick clouds, and and spins around its orbit so both sides get heat from the sun."

Of course, this loses a lot of the detail justifying the descriptions, such as what chemicals are making the planet blue or what temperature causes the star to be red, but it could resolve the inconsistency errors seen in the tested models. The research within this paper has proven the simplest description is the best.

The most important lesson, however, is that developing this model and method of generative art is a promising and compelling field of research. Not only has this method bridged art and science and utilized the best of each to enhance the best of the other, but it has also allowed the visualization of something incapable of viewing, even with today's most advanced technology. The model is imperfect but a viable pursuit for further researchers and artists.

10 Ethical Concerns

NASA's Exoplanet dataset and Image and Video Library used for this project are open source and do not contain any personal, private, or secure information that could be potentially unethical. The models and training of these models involve ethical concerns that require acknowledgment and consideration for future use in this work.

In April 2023, a lawsuit was filed against Stable Diffusion for violating copyright protections of visual artists (Vincent, 2023). These artists claim Stable Diffusion copied 12 million pieces of artwork without permission or compensation to the rightful owners. A resolution has not been reached, but should Stable Diffusion be found guilty, further use of this research could result in the endorsement of unethical and illegal business practices. This may not be a current concern for those interested in this research but should be taken seriously for future iterations and applications.

In addition, training a generative model like Stable Diffusion is extremely costly financially and in terms of computing power. To conduct this project required Google Colab's Nvidia A100 and V100 GPUs, which were connected remotely. On average, it took between four and five hours to fine-tune the model with a dataset of 127 image and prompt pairs. This duration includes using a LoRA model, which reduced computing power from 30 GB of RAM to 15.7 GB of RAM. To gain access to Colab's GPUs for this length of time, across four training iterations, required a Google Colab Pro+ account, which costs forty-five pounds per month. Though you could train a smaller model with a free account or Pro account at just ten pounds per month, to repeat and grow on this research would require a similar expense. This expense limits access to those with the funds necessary to conduct the research, therefore, limiting the potential applications of this work.

Furthermore, running these GPUs consumes a significant amount of energy. It is estimated that training an AI model, such as Stable Diffusion, could produce 626,000 pounds of CO2 (Strubell, E, 2019). Though fine-tuning a model produces less CO2 than training one, it is necessary to consider the environmental effects of reproducing and building on this type of research, particularly should it be expanded upon with larger datasets or trained for longer durations.

11 Conclusion

This project set out to fine-tune Stable Diffusion with generated datasets created using NASA's Image and Video Library and NASA's Exoplanet dataset. In fine-tuning Stable Diffusion, the project emphasized developing prompts that adhered to the mathematical and scientific values within the data. By training a model on these prompts, the generated images are no longer art but predicted visualizations of the exoplanet.

Acquiring enough research to create successful prompts was a challenge of this project and required a significant portion of the project's timeline. This challenge was coupled with the difficulty of obtaining a successfully trained Stable Diffusion model. The first iteration of training with Stable Diffusion 2.1 failed despite different attempts to achieve successful results. The second iteration of training with Stable Diffusion 1.5 was far more successful and allowed for five training sessions. Two training sessions used a maximum of 77 tokens, trained

at 10 and 70 epochs. The last three training sessions used a maximum of 288 tokens, trained at 70 epochs per session. Though all four training sessions at 70 epochs successfully generated images for the selected prompts, some were better at adhering to the given prompts than others. Between the three models trained with 288 tokens, the mass_prompt proved more consistently accurate than the ratio_prompt and the size_text_prompt. The ratio_prompt and size_text_prompt failed to adhere to the prompts and to produce consistent, realistic images. Many images produced by both models appeared almost cartoon-like, with unnatural backgrounds and combinations of planets (Figures 33 and 34). The images generated by the mass_prompt model did not closely adhere to the input prompt but replicated the realism depicted in the input images and desired from the project results (Figure 37).

The best model was the Stable Diffusion 1.5 model trained at 70 epochs with a maximum of 77 tokens. This model consistently visualized the tested prompts closely to the desired style of the input images within the four universally tested prompts and within the additional testing (Figures 17-20, Figure 35, Figure 36). Though all models failed to perfectly adhere to the desired prompts one hundred percent of the time, the results confirm the shorter prompt is far more successful than its counterparts in producing realistic images of these distant worlds.

Given the failed training with Stable Diffusion 2.1, the accuracy of the images produced by the successfully trained models with Stable Diffusion 1.5 proves the validity of using such a method to visualize and represent scientific research. This project, however, could be improved and encourages those in the field of astrophysics or other fields of science to build upon and expand the scope of this work.

12 Recommendations

When developing the idea for this project, the challenges faced could have been little predicted. The most significant of these was the technical and research-based complications - such as version control issues when fine-tuning Stable Diffusion, machine compatibility issues with Stable Diffusions' packages and requirements, and acquiring enough scientific basis to develop research-based prompts - that caused disruptions in the project's timeline and limitations in the project's results. It is not expected any form of preparation could have prepared for or resolved these issues for this iteration of the research. For future iterations, the below recommendations are made:

First, collaborate with astrophysicists and researchers to review the materials provided and suggest improvements. The research conducted for this project was extensive, and the decisions made to create the prompts and descriptors for each planet and star type were made on widely accepted research. That said, what determines a planet's visual characteristics can be any number of factors only confirmed with spectral imaging or conducting spectroscopy. Many exoplanets within the Exoplanet dataset are too far away for either option with today's technology. Having a team of astrophysicists and researchers to offer their suggestions and experiences to improve the prompts could increase the accuracy of the model's predicted visualizations.

Second, test a variety of textual encoders and prompt structures. This was not attempted due to a limitation in computing power and compute units, but altering the text encoder

and prompt structure could be an interesting form of research that may increase the overall accuracy of the fine-tuned model. Though this model proved accurate most of the time, there are faults in the model's adherence to the prompts generated. The selected format of the prompts or the textual encoder may be disrupting the computer's interpretation of the prompt. By testing a variety of encoders and prompt structures, this inaccuracy could be resolved or attributed to another feature than the text encoder.

Third, increase the size of the dataset. The dataset used includes 127 image and prompt pairs. This dataset was created directly from NASA's Image and Video Library and only includes images from that Library. This decision was made to validate the scientific qualities of the image-prompt pairs, but it would be interesting to pursue the effects of increasing the dataset size on the model's accuracy. Perhaps increasing the dataset to 700, close to the Hugginface model, might increase the success in fine-tuning and producing consistently reliable results. It is worth mentioning that using images generated by other models reduces the ability to determine a scientific background for the image's prompts. This may or may not be detrimental to training but affects the scientific basis of this research. Depending on what the model is used for if attempting to train the model with more images, it should be considered what the purposes are for training and where the additional images are coming from; artists may not require as scientific a basis as a researcher developing scientific models might.

Fourth, utilize a local GPU with at least 30 GB of RAM on a Linux or Windows-based machine. This research used Google Colab as Mac OS is not compatible with several of the Stable Diffusion dependencies. Google Colab allows the connection to a remote GPU, including the Nvidia A100 and Nvidia V100. Both grants the user access to 30GB of RAM. The caveat to this, however, is the limitation to how much run time is available and how long a run time can last. Colab only allows 100 compute units for Pro members and 500 for Pro + members. Running the Nvidia A100 ranges from 5-8 compute units per hour, and approximately 4 hours and 15 minutes to run 70 epochs through a dataset of 127 image and prompt pairs. This runtime includes a LoRA model, which "accelerates the training of large models while consuming less memory" (Stable Diffusion). In addition, there is a 12-hour limitation on training within the Colab environment, which means that after 12 hours, training resets, and all progress is lost. With a local GPU of 30 GB of RAM, more experiments could occur on a larger dataset without concern or limitation of using all available training resources.

With these recommendations, future iterations of this research could secure more accurate depictions of these distant worlds. The intention of developing this model was, in part, to establish a link between the scientific and the creative. Though this model is limited by the scientific data available, it provides a strong foundation for those interested in developing a scientifically based generative model for envisioning distant worlds, and perhaps for those in other fields on the applications of generative art within the scientific community.

13 Supplemental Materials

Figure 38: Test Prompts

Model	Prompt 9	Prompt 14	Prompt 488	Prompt 687
75_tokens	“ A light orange, small star with a massive, gas-giant planet. The planet a shade of red in color, has soft-edged stripes of thick clouds, and spins around its orbit so both sides get heat from the sun”	“An orange red, very small star with a massive, gas-giant planet. The planet is mostly blue with traces of white coloring, has clear, sharp-edge stripes of thick clouds, and spins around its orbit so both sides get heat from the sun”	“A yellow, medium small star with a massive, gas-giant planet. The planet is mostly blue with traces of white coloring, has clear, sharp-edge stripes of thick clouds, and spins around its orbit so both sides get heat from the sun”	A light orange, small star with a rocky, super-earth planet. The planet is a varying shade of blue and/or white, is hot and rotating quickly with little to no clouds, and only has one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold”
mass_prompt	“A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This gas-giant planet is 3000 times the size of earth, is dominated by silicate and iron clouds most notably variations of red coloring and has stripes of thick clouds of various coloring defined by softened edges. This planet spins around its orbit so both sides get heat from the sun. The planet’s star is light orange and an unknown size compared to the size of the sun.”	“A solar system made up of 1 planet(s), 2 star(s), and 0 moon(s). This gas-giant planet is 4131 times the size of earth, likely has water vapor producing a blue color as well as helium and hydrogen, which both produce shades of white and has stripes of thick clouds of various coloring defined by clear, sharp edges. This planet spins around its orbit so both sides get heat from the sun. The planet’s star is orange red and an unknown size compared to the size of the sun.”	“A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This gas-giant planet is 3623 times the size of earth, likely has water vapor producing a blue color as well as helium and hydrogen, which both produce shades of white and has stripes of thick clouds of various coloring defined by clear, sharp edges. This planet spins around its orbit so both sides get heat from the sun. The planet’s star is yellow and the same size as the sun.”	“A solar system made up of 2 planet(s), 1 star(s), and 0 moon(s). This super-earth planet is 5 times the size of earth, has carbon dioxide and hydrocarbons are dominant in this planet which could both come in varying shades of blue and white and is hot and rotating quickly with little to no atmosphere, clouds, or storms. This planet only has one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold. The planet’s star is light orange and an unknown size compared to the size of the sun.”
ratio_prompt	“A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This gas-giant planet is dominated by silicate and iron clouds most notably variations of red coloring and has stripes of thick clouds of various coloring defined by softened edges. This planet spins around its orbit so both sides get heat from the sun. The planet’s star is light orange and 0.02833333412806193 the size of its planet.”	“A solar system made up of 1 planet(s), 2 star(s), and 0 moon(s). This gas-giant planet likely has water vapor producing a blue color as well as helium and hydrogen, which both produce shades of white and has stripes of thick clouds of various coloring defined by clear, sharp edges. This planet spins around its orbit so both sides get heat from the sun. The planet’s star is orange red and 0.004598491109652925 the size of its planet.”	“A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This gas-giant planet likely has water vapor producing a blue color as well as helium and hydrogen, which both produce shades of white and has stripes of thick clouds of various coloring defined by clear, sharp edges. This planet spins around its orbit so both sides get heat from the sun. The planet’s star is yellow and 0.03836341205303896 the size of its planet.”	“A solar system made up of 2 planet(s), 1 star(s), and 0 moon(s). This super-earth planet has carbon dioxide and hydrocarbons are dominant in this planet which could both come in varying shades of blue and white and is hot and rotating quickly with little to no atmosphere, clouds, or storms. This planet only has one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold. The planet’s star is light orange and 13.982301028727623 the size of its planet.”
size_text_prompt	“A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This massive gas-giant planet is dominated by silicate and iron clouds most notably variations of red coloring. This planet has stripes of thick clouds of various coloring defined by softened edges. The planet’s star is spins around its orbit so both sides get heat from the sun and light orange.”	“A solar system made up of 1 planet(s), 2 star(s), and 0 moon(s). This massive gas-giant planet likely has water vapor producing a blue color as well as helium and hydrogen, which both produce shades of white. This planet has stripes of thick clouds of various coloring defined by clear, sharp edges. The planet’s star is spins around its orbit so both sides get heat from the sun and orange red.”	“A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This massive gas-giant planet likely has water vapor producing a blue color as well as helium and hydrogen, which both produce shades of white. This planet has stripes of thick clouds of various coloring defined by clear, sharp edges. The planet’s star is spins around its orbit so both sides get heat from the sun and yellow.”	“A solar system made up of 2 planet(s), 1 star(s), and 0 moon(s). This rocky super-earth planet has carbon dioxide and hydrocarbons are dominant in this planet which could both come in varying shades of blue and white. This planet is hot and rotating quickly with little to no atmosphere, clouds, or storms. The planet’s star is only having one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold and light orange.”

Figure 39: Additional Test Prompts

Model	Prompt Number	Prompt Text
ratio_prompt	Prompt 3219	"A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This terrestrial planet is 1 time the size of earth, has carbon dioxide and hydrocarbons are dominant in this planet which could both come in varying shades of blue and white and is hot and rotating quickly with little to no atmosphere, clouds, or storms. This planet only has one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold. The planet's star is yellow and 57.55395828446681 the size of its planet."
size_text_prompt	Prompt 4594	"A solar system made up of 1 planet(s), 1 star(s), and 0 moon(s). This Neptune-like planet methane is breaking down and possibly mixing with other chemicals such as sulfur, known for its pale-yellow color. This planet has clearly defined striped light and dark icy clouds. The planet's star is only having one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold and yellow."
75_tokens	Prompt 1227	"A yellow white, medium star with a nan, super-earth planet. The planet is white and pale yellow in color, is hot and rotating quickly with little to no clouds, and only has one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold"
75_tokens	Prompt 1066	"A white, large star with a Neptune-like planet. The planet is mostly blue mixing with brown and red colors, has clearly defined striped light and dark clouds, and only has one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold"
mass_prompt	Prompt 1168	"A solar system made up of 2 planet(s), 1 star(s), and 0 moon(s). This gas-giant planet is 1195 times the size of earth, likely has water vapor producing a blue color as well as helium and hydrogen, which both produce shades of white and has stripes of thick clouds of various coloring defined by clear, sharp edges. This planet only has one side of the planet facing the sun. The side facing the sun is extremely hot and the side that faces away from the sun is dark and cold. The planet's star is yellow and 1 time the size of the sun."

Figure 40: Training Parameters

1.1 Install Dependencies

2.1 Download Available Model

- model_name: AnyLoRA
- 2_model_name: Stable-Diffusion-2-1-base

3.1 Locating Data Directory

- train_data_dir: /content/LoRA/data

4.4 Bucketing and Latents Catching

- model_dir:/content/pretrained_model/Stable-Diffusion-2-1-base.safetensors
- input_json: /content/LoRA/metadata.json
- output_json: /content/LoRA/meta_lat.json
- batch_size: 8
- max_data_loader_n_workers: 2
- mixed_precision: fp16

5.1 Model Config

- project_name: exoplanet_data_sd_2.1
- pretrained_model_name_or_path: /content/pretrained_model/ Stable-Diffusion-2-1-base.safetensors
- output_dir: /content/LoRA/output
- output_to_drive: yes

5.2 Dataset Config

- dataset_repeats: 10
- in_json: /content/LoRA/metadata.json
- resolution: 512, 512
- keep_tokens: 0

5.3 LoRA and Optimizer Config

- network_category: LoRA
- conv_dim: 32
- conv_alpha: 0
- network_dim: 32
- network_alpha: 16
- min_snr_gamma: -1
- optimizer_type: AdamW8bit
- train_unet: yes
- unet_lr: 1e-4
- train_text_encoder: yes
- text_encoder_lr: 5e-5
- lr_scheduler: constant

5.4 Training Config

- low_ram: yes
- samler: ddim
- noise_offset: 1
- num_epochs: 10
- train_batch_size: 6
- mixed_precision: fp16
- save_precision: fp16
- save_n_epochs_type_value: 1
- save_model_as: safetensors
- max_token_length: 77
- clip_skip: 2
- gradiant_accumulation_steps: 1
- seed: -1

5.5 Start Training

- sample_prompt: /content/LoRA/config/sample_prompt.txt
- config_file: /content/LoRA/config/config_file.toml

Figure 41: Getting Images for Training Dataset

```
url = "http://images-api.nasa.gov/search?q=exoplanet/api_key="
api_key = "0nG0tpWyrzxaZS6k6atKgtw6KbqccB8qIAj9h5rv"

import nasapy #python wrapper for NASA API
from nasapy import media_search
import os #haven't used yet
import matplotlib.pyplot as plt
from skimage import io
import pandas as pd

nasa = nasapy.Nasa(key=api_key)

def get_images(database_name):
    for i, image in enumerate(database_name):
        link_data = image['links']

        for url in link_data:
            image_url = url['href']
            image_to_show = io.imread(image_url)
            plt.imshow(image_to_show)
            plt.xlabel(i)
            plt.show()

keep_images = []

def save_keep_images(database_name, keep_indexes):
    for i, image in enumerate(database_name):
        if i in keep_indexes:
            keep_images.append(database_name[i])
    return keep_images

exoplanet_images = media_search(query="planet artist concept", media_type="image") #
↪  repeated for additional queries
exoplanet_data = exoplanet_images["items"]

get_images(exoplanet_data)
keep_indexes = [4, 6, 14, 29, 77, 80, 87]
save_keep_images(exoplanet_data, keep_indexes)

exoplanet_image_data = pd.DataFrame.from_dict(keep_images)

exoplanet_image_data = exoplanet_image_data.explode('data')
exoplanet_image_data = exoplanet_image_data.explode('links')

exoplanet_image_data = exoplanet_image_data.drop(['href'], axis=1)
```

```
image_description = []
for d in exoplanet_image_data['data']:
    image_description.append(d['description'])

exoplanet_image_data['image_description'] = image_description

image_link = []
for l in exoplanet_image_data['links']:
    print(l['href'])
    image_link.append(l['href'])

exoplanet_image_data['image_link'] = image_link

exoplanet_image_data = exoplanet_image_data.drop(["links", "data"], axis=1)

exoplanet_image_data.to_csv("exoplanet_image_data.csv")
```

Figure 42: Getting Stellar Color

```
def get_stellar_color(dataset):
    for index, data in dataset.iterrows():
        if data['st_spectype'] != 0:
            #harvard standard spectral classifications
            if data['st_spectype'][0] == 'M' or data['st_spectype'][0] == 'm':
                dataset.at[index, 'stellar_color'] = 'orange red'
            elif data['st_spectype'][0] == 'K':
                dataset.at[index, 'stellar_color'] = 'light orange'
            elif data['st_spectype'][0] == 'G':
                dataset.at[index, 'stellar_color'] = 'yellow'
            elif data['st_spectype'][0] == 'F':
                dataset.at[index, 'stellar_color'] = 'yellow white'
            elif data['st_spectype'][0] == 'A':
                dataset.at[index, 'stellar_color'] = 'white'
            elif data['st_spectype'][0] == 'B':
                dataset.at[index, 'stellar_color'] = 'blue white'
            elif data['st_spectype'][0] == 'O':
                dataset.at[index, 'stellar_color'] = 'blue'
            #special case star classifications
            elif data['st_spectype'][0] == 'T': #late stage brown dwarf
                dataset.at[index, 'stellar_color'] = 'violet'
            elif data['st_spectype'][0] == 'L': #early stage brown dwarf
                dataset.at[index, 'stellar_color'] = 'magenta'
            elif data['st_spectype'] == 'WD' or data['st_spectype'][0] == 'D': #white
            ↪  dwarf
                dataset.at[index, 'stellar_color'] = 'white'
```

```
        elif data['st_spectype'][0] == 's':
            if data['st_teff'] <= 3500.0:
                dataset.at[index, 'stellar_color'] = 'orange red'
            elif 3500.0 < data['st_teff'] <= 5000.0:
                dataset.at[index, 'stellar_color'] = 'light orange'
            elif 5000.0 < data['st_teff'] <= 6000.0:
                dataset.at[index, 'stellar_color'] = 'yellow'
            elif  6000.0 < data['st_teff'] <= 7500.0:
                dataset.at[index, 'stellar_color'] = 'yellow white'
            elif 7500.0 < data['st_teff'] <= 11000.0:
                dataset.at[index, 'stellar_color'] = 'white'
            elif 11000.0 < data['st_teff'] <= 25000.0:
                dataset.at[index, 'stellar_color'] = 'blue white'
            elif 25000.0 < data['st_teff'] <= 100000.0:
                dataset.at[index, 'stellar_color'] = 'blue'
            elif 100000.0 < data['st_teff']:
                dataset.at[index, 'stellar_color'] = 'white'
    elif data['st_spectype'] == 0:
        if data['st_teff'] <= 3500.0:
            dataset.at[index, 'stellar_color'] = 'orange red'
        elif 3500.0 < data['st_teff'] <= 5000.0:
            dataset.at[index, 'stellar_color'] = 'light orange'
        elif 5000.0 < data['st_teff'] <= 6000.0:
            dataset.at[index, 'stellar_color'] = 'yellow'
        elif  6000.0 < data['st_teff'] <= 7500.0:
            dataset.at[index, 'stellar_color'] = 'yellow white'
        elif 7500.0 < data['st_teff'] <= 11000.0:
            dataset.at[index, 'stellar_color'] = 'white'
        elif 11000.0 < data['st_teff'] <= 25000.0:
            dataset.at[index, 'stellar_color'] = 'blue white'
        elif 25000.0 < data['st_teff'] <= 100000.0:
            dataset.at[index, 'stellar_color'] = 'blue'
        elif 100000.0 < data['st_teff']:
            dataset.at[index, 'stellar_color'] = 'white'

    elif data['st_spectype'] == 0 & data['st_teff'] == 0:
        if data['st_mass'] <= 0.45:
            dataset.at[index, 'stellar_color'] = 'orange red'
        elif 0.45 < data['st_mass'] <= 0.8:
            dataset.at[index, 'stellar_color'] = 'light orange'
        elif 0.8 < data['st_mass'] <= 1.04:
            dataset.at[index, 'stellar_color'] = 'yellow'
        elif  1.04 < data['st_mass'] <= 1.4:
            dataset.at[index, 'stellar_color'] = 'yellow white'
        elif 1.4 < data['st_mass'] <= 2.1:
            dataset.at[index, 'stellar_color'] = 'white'
        elif 2.1 < data['st_mass'] <= 16:
            dataset.at[index, 'stellar_color'] = 'blue white'
        elif 16 < data['st_mass']:
            dataset.at[index, 'stellar_color'] = 'blue'

    else:
        dataset.at[index, 'stellar_color'] = 'unknown stellar color'

return dataset
```

Figure 43: Getting Stellar Size as Text

```
def stellar_mass_description(dataset):
    for index, data in dataset.iterrows():
        st_spectype = str(data['st_spectype'])
        if st_spectype[0] == 'W' or st_spectype[0] == 'D' or st_spectype[0] == 'L' or
        ↪  st_spectype[0] == 'T' or st_spectype[0] == 's':
            dataset.at[index, 'stellar_mass_description'] = 'tiny'
        elif data['stellar_color'] == 'orange red':
            dataset.at[index, 'stellar_mass_description'] = 'very small'
        elif data['stellar_color'] == 'light orange':
            dataset.at[index, 'stellar_mass_description'] = 'small'
        elif data['stellar_color'] == 'yellow':
            dataset.at[index, 'stellar_mass_description'] = 'medium small'
        elif data['stellar_color'] == 'yellow white':
            dataset.at[index, 'stellar_mass_description'] = 'medium'
        elif data['stellar_color'] == 'white':
            dataset.at[index, 'stellar_mass_description'] = 'large'
        elif data['stellar_color'] == 'blue white':
            dataset.at[index, 'stellar_mass_description'] = 'giant'
        elif data['stellar_color'] == 'blue':
            dataset.at[index, 'stellar_mass_description'] = 'massive'
        else:
            dataset.at[index, 'stellar_mass_description'] = 'unknown size'
    return dataset
```

Figure 44: Chemical Properties

Chemical	Solid Temperature (Kelvin, K)	Liquid Temperature (K)	Gas Temperature (K)	Color Properties
Helium	<0.95	0.95	4.2	Helium is colorless, but reflects visible light
Hydrogen	<14.01	14.04	20.28	Hydrogen is colorless, but reflects visible light
Nitrogen	<63.17	63.17	77.36	Nitrogen is colorless, but reflects visible light
Methane	<90.694	90.694	111.65	Methane absorbs red light and reflects blue light
Oxygen	<54.36	54.36	90.188	Liquid oxygen, absorbs red light and reflects blue light, Oxygen as a gas scatters blue and violet wavelengths, causing a blue appearance
Ammonia	<195.4	195.4	239.81	Ammonia crystals, solid ammonia, and ammonia compounds range from light yellow to brown in color
Hydrogen Deuteride	<14.1	14.1	20.11	Hydrogen Deuteride is colorless, but reflects visible light
Ethane	<90.4	90.4	184.6	Ethane is colorless, but reflects starlight in the ultra-violet
Water	<273	273	373	Water absorbs red light and reflects blue light
Carbon Dioxide	<195			Carbon Dioxide exists solely as a gas and solid under our knowable environment. Carbon Dioxide is colorless, but absorbs infrared radiation
Argon	<83.8	83.8	87.3	Argon is colorless and does not interact with visible or infrared light
Carbon Monoxide	<63.18	63.18	81.6	Carbon Monoxide is colorless but absorbs infrared radiation
Nitrogen Oxide	<109	109	121	Nitrogen Oxide is reddish-brown in color as a gas and brown in a liquid and solid state
Neon	<24.56	24.56	27.1	Neon is a colorless gas but appears orange-red when excited by an electric charge
Krypton	<115.79	115.79	119.75	Krypton as a solid is white, but as a liquid and gas it appears green and yellow when excited by an electric charge
Xenon	<161.4	161.4	165.03	Xenon is colorless, but emits blue when excited by an electric charge
Sulfur Dioxide	<201	201	263	Sulfur Dioxide gas is colorless but reflects visible light
Sodium	<370	370	1894.15	Sodium is silvery-white in color in all states of matter

Figure 45: Getting Planet Category

```
def get_planet_category(dataset):
    if dataset['pl_bmasse'].any() != 0.0:
        conditions = [(dataset['pl_bmasse'] > 0.0) & (dataset['pl_bmasse'] <= 2.0),
                      (dataset['pl_bmasse'] > 2.0) & (dataset['pl_bmasse'] <= 10.0),
                      (dataset['pl_bmasse'] > 10.0) & (dataset['pl_bmasse'] <= 17.0),
                      (dataset['pl_bmasse'] > 17.0),
                      (dataset['pl_bmasse'] == 0.0)
                      ]

        values = ['terrestrial', 'super-earth', 'neptune-like', 'gas-giant', 'unknown
        ↪  planet size']

        dataset['planet_category'] = np.select(conditions, values, default='unknown')

    elif dataset['pl_bmasse'].any() == 0.0 and dataset['pl_rade'] != 0.0:
        conditions = [(dataset['pl_rade'] > 0.0) & (dataset['pl_rade'] <= 2.0),
                      (dataset['pl_rade'] > 2.0) & (dataset['pl_rade'] <= 10.0),
                      (dataset['pl_rade'] > 10.0) & (dataset['pl_rade'] <= 17.0),
                      (dataset['pl_rade'] > 17.0),
                      (dataset['pl_rade'] == 0.0)
                      ]

        values = ['terrestrial', 'super-earth', 'neptune-like', 'gas-giant', 'unknown
        ↪  planet size']

        dataset['planet_category'] = np.select(conditions, values, default='unknown')

    return dataset
```

Figure 46: Getting Planet Size as Text

```
def planet_mass_description(dataset):
    for index, data in dataset.iterrows():

        mercury_mass = 0.0553
        venus_mass = 0.815
        earth_mass = 1.0
        mars_mass = 0.107
        jupiter_mass = 317.8
        saturn_mass = 95.2
        uranus_mass = 14.5
        neptune_mass = 17.1

        if data['pl_bmasse'] != 0:
            if data['pl_bmasse'] <= mercury_mass:
                dataset.at[index, 'planet_mass_description'] = 'tiny'
            elif mercury_mass < data['pl_bmasse'] <= mars_mass:
                dataset.at[index, 'planet_mass_description'] = 'very small'
            elif mars_mass < data['pl_bmasse'] <= venus_mass:
                dataset.at[index, 'planet_mass_description'] = 'small'
            elif venus_mass < data['pl_bmasse'] <= earth_mass:
                dataset.at[index, 'planet_mass_description'] = 'medium small'
            elif uranus_mass < data['pl_bmasse'] <= neptune_mass:
                dataset.at[index, 'planet_mass_description'] = 'medium'
            elif neptune_mass < data['pl_bmasse'] <= saturn_mass:
                dataset.at[index, 'planet_mass_description'] = 'large'
            elif saturn_mass < data['pl_bmasse'] <= jupiter_mass:
                dataset.at[index, 'planet_mass_description'] = 'giant'
            elif jupiter_mass < data['pl_bmasse']:
                dataset.at[index, 'planet_mass_description'] = 'massive'

        elif data['pl_bmasse'] == 0:
            if data['planet_category'] == 'terrestrial':
                dataset.at[index, 'planet_mass_description'] = 'small'
            elif data['planet_category'] == 'super-earth':
                dataset.at[index, 'planet_mass_description'] = 'medium'
            elif data['planet_category'] == 'neptune-like':
                dataset.at[index, 'planet_mass_description'] = 'large'
            elif data['planet_category'] == 'gas-giant':
                dataset.at[index, 'planet_mass_description'] = 'giant'

        else:
            dataset.at[index, 'planet_mass_description'] = 'unknown size'

    return dataset
```

Figure 47: Getting Planet Color

```
def get_planet_description(dataset):
    for index, data in dataset.iterrows():
        mercury_mass = 0.0553
        venus_mass = 0.815
        earth_mass = 1.0
        mars_mass = 0.107
        jupiter_mass = 317.8
        saturn_mass = 95.2
        uranus_mass = 14.5
        neptune_mass = 17.1

        if data['pl_eqt'] != 0.0:
            if data['planet_category'] == 'terrestrial' or data['planet_category'] ==
            ↪  'super-earth':
                if data['pl_eqt'] <= 20.0:
                    dataset.at[index, 'planet_color'] = 'has a composition of hydrogen
                    ↪  and helium producing a distince white color'
                elif 20.0 < data['pl_eqt'] <= 200.0:
                    dataset.at[index, 'planet_color'] = 'has high quantities of methane
                    ↪  known for its rich blue color'
                elif 200.0 < data['pl_eqt'] <= 400.0:
                    dataset.at[index, 'planet_color'] = 'likely has a small amount of
                    ↪  blue methane and yellow ammonia. The most dominant color would
                    ↪  come from blue liquid water'
                elif 400.0 < data['pl_eqt'] <= 600.0:
                    dataset.at[index, 'planet_color'] = 'most likely has water vapor
                    ↪  that still produces a true blue color mixing with the breakdown
                    ↪  of methanes deep blue'
                elif 600.0 < data['pl_eqt'] <= 800.0:
                    dataset.at[index, 'planet_color'] = 'has carbon dioxide and
                    ↪  hydrocarbons are dominant in this planet which could both come
                    ↪  in varying shades of blue and white'
                elif 800.0 < data['pl_eqt'] <= 1200.0:
                    dataset.at[index, 'planet_color'] = 'has white carbon dioxide
                    ↪  molecules and pale yellow sulfur compounds are likely on this
                    ↪  planet'
                elif 1200.0 < data['pl_eqt'] <= 1700.0:
                    dataset.at[index, 'planet_color'] = 'has pale yellow sulfure
                    ↪  compounds and blue and white water vapor are likely dominate on
                    ↪  this planet'
                elif 1700.0 < data['pl_eqt']:
                    dataset.at[index, 'planet_color'] = 'is so hot all metals are
                    ↪  breaking down causing the planet to likely be covered in lava'
            elif data['planet_category'] == 'neptune-like':
                if data['pl_eqt'] <= 90.0:
                    dataset.at[index, 'planet_color'] = 'consists mostly of helium and
                    ↪  hydrogen which are dominantly white, but it mixes with frozen
                    ↪  methane characterized by a light blue color'
```

```
        elif 90.0 < data['pl_eqt'] <= 110.0:
            dataset.at[index, 'planet_color'] = 'has methane as a liquid and
            ↪  dominant in the atmosphere shifting the color to a azure blue
            ↪  color'
        elif 110.0 < data['pl_eqt'] <= 275.0:
            dataset.at[index, 'planet_color'] = 'has methane as a gas and
            ↪  producing a deep blue color'
        elif 275.0 < data['pl_eqt'] <= 375.0:
            dataset.at[index, 'planet_color'] = 'has a dark blue methane color
            ↪  mixing with water vapor clouds of a much lighter blue color and
            ↪  traces of ammonia as a light yellow color'
        elif 375.0 < data['pl_eqt'] <= 500.0:
            dataset.at[index, 'planet_color'] = 'methane is breaking down and
            ↪  possibly mixing with other chemicals such as sulfur, known for
            ↪  its pale yellow color'
        elif 500.0 < data['pl_eqt'] <= 800.0:
            dataset.at[index, 'planet_color'] = 'methane is breaking down, so
            ↪  the planet is likely no longer a deep blue, but hydrocarbons are
            ↪  likely present in the atmosphere, which depending on composition
            ↪  are varying shades of blue'
        elif 800.0 < data['pl_eqt'] <= 900.0:
            dataset.at[index, 'planet_color'] = 'has deep blue methane is
            ↪  breaking down and less pronounced and likely to have alkali
            ↪  metals known for their silvery white color'
        elif 900.0 < data['pl_eqt'] <= 1400.0:
            dataset.at[index, 'planet_color'] = 'has deep blue methane is
            ↪  breaking down and less pronounced, aerosols and thermal
            ↪  emissions are more likely and often give off a neutral or red
            ↪  color that would mix with the blue'
        elif 1400.0 < data['pl_eqt']:
            dataset.at[index, 'planet_color'] = 'likely overtaken by aerosols
            ↪  and thermal emissions as well as high-temperature gases causing
            ↪  it to be between purple and red in color'
    elif data['planet_category'] == 'gas-giant':
        if data['pl_eqt'] <= 70.0:
            dataset.at[index, 'planet_color'] = 'has frozen ammonia producing a
            ↪  duller yellow color merging with the more dominant methane,
            ↪  characterized by its shade of blue'
        elif 70.0 < data['pl_eqt'] <= 150.0:
            dataset.at[index, 'planet_color'] = 'most likely overrun with
            ↪  ammonia clouds characterized by their variety of yellow coloring'
        elif 150.0 < data['pl_eqt'] <= 250.0:
            dataset.at[index, 'planet_color'] = 'has methane in its blue color
            ↪  but in very small quantities. The dominant color will be
            ↪  ammonia, which is now a liquid giving the planet a darker yellow
            ↪  color closer to brown'
         elif 250.0 < data['pl_eqt'] <= 350.0:
            dataset.at[index, 'planet_color'] = 'the atmosphere is overtaken
            ↪  with water vapor giving the planet a mostly white color with the
            ↪  posibility of slight blue tinting'
```

```
        elif 350.0 < data['pl_eqt'] <= 800.0:
            dataset.at[index, 'planet_color'] = 'is so warm it likely does not
            ↪  have clouds and appears as a uniform blue orb'
        elif 800.0 < data['pl_eqt'] <= 900.0:
            dataset.at[index, 'planet_color'] = 'is in transition from a blue
            ↪  atmosphere to being overtaken by carbon monoxide and alkali
            ↪  metals known for being silvery white'
        elif 900.0 < data['pl_eqt']<= 1400.0:
            dataset.at[index, 'planet_color'] = 'has carbon monoxide and alkali
            ↪  metals like sodium and potassium as dominant, which known for
            ↪  their silvery white coloring'
        elif 1400.0 < data['pl_eqt']:
            dataset.at[index, 'planet_color'] = 'is dominated by silicate and
            ↪  iron clouds most notably variations of red coloring'
elif data['planet_category'] == 'unknown planet size':
        dataset.at[index, 'planet_color'] = 'is an unknown planet color'
    elif data['pl_eqt'] == 0.0 and data['pl_bmasse'] != 0.0:
    if data['pl_bmasse'] <= mercury_mass:
        dataset.at[index, 'planet_color'] = 'is likely extremely hot and
        ↪  possibly covered in lava, primary composed of silicate minerals and
        ↪  oxides ranging in a variety of colors from silvery gray to a rich
        ↪  deep red'
    elif mercury_mass < data['pl_bmasse'] <= mars_mass:
        dataset.at[index, 'planet_color'] = 'is primarily composed of silicate
        ↪  minerals and oxides ranging in a variety of colors from silvery gray
        ↪  to a rich deep red'
    elif mars_mass < data['pl_bmasse'] <= venus_mass:
        dataset.at[index, 'planet_color'] = 'is primarily composed of silicate
        ↪  minerals and oxides ranging in a variety of colors from silvery gray
        ↪  to a rich deep red, as well as other gas chemicals such as carbon
        ↪  dioxide which produces a white color, and sulfur known for being a
        ↪  pale yellow'
    elif venus_mass < data['pl_bmasse'] <= earth_mass:
        dataset.at[index, 'planet_color'] = 'likely has a mixture of blue liquid
        ↪  water, and other gas chemicals such as carbon dioxide which produces
        ↪  a white color, and sulfur known for being pale yellow in color'
    elif earth_mass < data['pl_bmasse']:
        dataset.at[index, 'planet_color'] = 'likely has water vapor producing a
        ↪  blue color as well as helium and hydrogen, which both produce shades
        ↪  of white'
    elif data['pl_bmasse'] <= uranus_mass:
        dataset.at[index, 'planet_color'] = 'consisting mostly of helium and
        ↪  hydrogen which are dominantly white, but it mixes with frozen
        ↪  methane characterized by a light blue color'
    elif uranus_mass < data['pl_bmasse'] <= neptune_mass:
        dataset.at[index, 'planet_color'] = 'has methane as a liquid and
        ↪  dominant in the atmosphere shifting the color to a azure blue color'
    elif neptune_mass < data['pl_bmasse']:
        dataset.at[index, 'planet_color'] = 'has methane as a gas and producing
        ↪  a deep blue color'
    elif data['pl_bmasse'] <= saturn_mass:
        dataset.at[index, 'planet_color'] = 'has frozen ammonia producing a
        ↪  duller yellow color with slight traces of methane characterized by
        ↪  its shade of true blue'
```

```
            elif saturn_mass < data['pl_bmasse'] <= jupiter_mass:
                dataset.at[index, 'planet_color'] = 'is most likely overrun with ammonia
                ↪  clouds characterized by their variety of yellow coloring'
            elif jupiter_mass < data['pl_bmasse']:
                dataset.at[index, 'planet_color'] = 'has methane in its blue color but
                ↪  in very small quantities with the dominant color will be ammonia,
                ↪  which is now a liquid giving the planet a range of darker yellow and
                ↪  brown colors'

        else:
            if data['planet_category'] == 'terrestrial' or data['planet_category'] ==
            ↪  'super-earth':
                dataset.at[index, 'planet_color'] = 'a rocky world made up of metals and
                ↪  rocks'
            elif data['planet_category'] == 'neptune-like':
                dataset.at[index, 'planet_color'] = 'an icy world composed of frozen
                ↪  gases'
            elif data['planet_category'] == 'gas-giant':
                dataset.at[index, 'planet_color'] = 'a giant world obscured by swirling
                ↪  gases'
            elif data['planet_category'] == 'unknown planet size':
                dataset.at[index, 'planet_color'] = 'has an unknown planet color'

    return dataset
```

Figure 48: Getting Planet Spin

```
def get_planet_spin(dataset):
    for index, data in dataset.iterrows():
        #planet orbit periods
        mercury_spin = 88
        venus_spin = 224
        earth_spin = 365
        mars_spin = 687
        jupiter_spin = 4332
        saturn_spin = 10747
        uranus_spin = 30589
        neptune_spin = 59800

        #planet masses
        mercury_mass = 0.0553
        venus_mass = 0.815
        earth_mass = 1.0
        mars_mass = 0.107
        jupiter_mass = 317.8
        saturn_mass = 95.2
        uranus_mass = 14.5
        neptune_mass = 17.1
```

```
if data['pl_orbper'] != 0.0:
    if data['planet_category'] == 'terrestrial' or data['planet_category'] ==
    ↪  'super-earth':
        if data['pl_orbper'] <= mercury_spin:
            dataset.at[index, 'planet_spin'] = 'is hot and rotating quickly with
            ↪  little to no atmosphere, clouds, or storms'
        elif mercury_spin < data['pl_orbper'] <= venus_spin:
            dataset.at[index, 'planet_spin'] = 'is hot and rotating quickly hot
            ↪  with a thick atmosphere of heavy swirling clouds with bright and
            ↪  dark markings'
        elif venus_spin < data['pl_orbper'] <= earth_spin:
            dataset.at[index, 'planet_spin'] = 'has clouds of various sizes
            ↪  speckling planet atmosphere showing pieces of the planet terrain
            ↪  beneath'
        elif earth_spin < data['pl_orbper'] <= mars_spin:
            dataset.at[index, 'planet_spin'] = 'has clouds of various sizes
            ↪  speckling planet atmosphere showing pieces of the planet terrain
            ↪  beneath'
        elif mars_spin < data['pl_orbper']:
            dataset.at[index, 'planet_spin'] = 'has wisps of clouds of various
            ↪  sizes speckling planet atmosphere showing most of the planet
            ↪  terrain beneath'
    elif data['planet_category'] == 'neptune-like':
        if data['pl_orbper'] <= uranus_spin:
            dataset.at[index, 'planet_spin'] = 'has clearly defined striped
            ↪  light and dark icy clouds'
        elif uranus_spin < data['pl_orbper'] <= neptune_spin:
            dataset.at[index, 'planet_spin'] = 'has softly defined striped light
            ↪  and dark icy clouds'
        elif neptune_spin < data['pl_orbper']:
            dataset.at[index, 'planet_spin'] = 'has icy clouds with no apparent
            ↪  delineation between colors'
    elif data['planet_category'] == 'gas-giant':
        if data['pl_orbper'] <= jupiter_spin:
            dataset.at[index, 'planet_spin'] = 'has stripes of thick clouds of
            ↪  various coloring defined by clear, sharp edges'
        elif jupiter_spin < data['pl_orbper'] <= saturn_spin:
            dataset.at[index, 'planet_spin'] = 'has stripes of thick clouds of
            ↪  various coloring defined by softened edges'
        elif saturn_spin < data['pl_orbper']:
            dataset.at[index, 'planet_spin'] = 'has thick clouds of various
            ↪  coloring blending together across the planet surface'
elif data['pl_orbper'] == 0.0:
    if data['planet_category'] == 'terrestrial' or data['planet_category'] ==
    ↪  'super-earth':
        if data['pl_bmasse'] <= mercury_mass:
            dataset.at[index, 'planet_spin'] = 'is hot and rotating quickly with
            ↪  little to no atmosphere, clouds, or storms'
        elif mercury_mass < data['pl_bmasse'] <= venus_mass:
            dataset.at[index, 'planet_spin'] = 'is hot and rotating quickly with
            ↪  a thick atmosphere of heavy swirling clouds with bright and dark
            ↪  markings'
        elif venus_mass < data['pl_bmasse'] <= earth_mass:
            dataset.at[index, 'planet_spin'] = 'has clouds of various sizes
            ↪  speckling planet atmosphere showing pieces of the planet terrain
            ↪  beneath'
```

```
            elif earth_mass < data['pl_bmasse'] <= mars_mass:
                dataset.at[index, 'planet_spin'] = 'has clouds of various sizes
                ↪  speckling planet atmosphere showing pieces of the planet terrain
                ↪  beneath'
            elif mars_mass < data['pl_bmasse']:
                dataset.at[index, 'planet_spin'] = 'has wisps of clouds of various
                ↪  sizes speckling planet atmosphere showing most of the planet
                ↪  terrain beneath'
        elif data['planet_category'] == 'neptune-like':
            if data['pl_bmasse'] <= uranus_mass:
                dataset.at[index, 'planet_spin'] = 'has clearly defined striped
                ↪  light and dark icy clouds'
            elif uranus_mass < data['pl_bmasse'] <= neptune_mass:
                dataset.at[index, 'planet_spin'] = 'has softly defined striped light
                ↪  and dark icy clouds'
            elif neptune_mass < data['pl_bmasse']:
                dataset.at[index, 'planet_spin'] = 'has icy clouds with no apparent
                ↪  delineation between colors'
        elif data['planet_category'] == 'gas-giant':
            if data['pl_bmasse'] <= jupiter_mass:
                dataset.at[index, 'planet_spin'] = 'has stripes of thick clouds of
                ↪  various coloring defined by clear, sharp edges'
            elif jupiter_mass < data['pl_bmasse'] <= saturn_mass:
                dataset.at[index, 'planet_spin'] = 'has stripes of thick clouds of
                ↪  various coloring defined by softened edges'
            elif saturn_mass < data['pl_bmasse']:
                dataset.at[index, 'planet_spin'] = 'has thick clouds of various
                ↪  coloring blending together across the planet surface'
        else:
            dataset.at[index, 'planet_spin'] = 'rotates around its star'

return dataset
```

Figure 49: Roche Limit and Tidal Locking

```
exoplanet_data['roche_limit'] = exoplanet_data.apply(lambda row:
    (1.26 * row.pl_rade * (row.st_mass / row.pl_bmasse) ** (1/3))
    if row.st_mass != 0 and row.pl_bmasse != 0 else 0, axis=1)

training_data['roche_limit'] = training_data.apply(lambda row:
    (1.26 * row.pl_rade * (row.st_mass / row.pl_bmasse) ** (1/3))
    if row.st_mass != 0 and row.pl_bmasse != 0 else 0, axis=1)

def tidal_locking(dataset):
    for index, data in dataset.iterrows():
        if data['pl_imppar'] != 0.0 or data['pl_orbsmax'] != 0.0:
            if data['pl_imppar'] or data['pl_orbsmax'] < data['roche_limit']:
                dataset.at[index, 'tidal_locked'] = 'only has one side of the planet
                ↪  facing the sun. The side facing the sun is extremely hot and the
                ↪  side that faces away from the sun is dark and cold'

            elif data['pl_imppar'] or data['pl_orbsmax'] >= data['roche_limit']:
                dataset.at[index, 'tidal_locked'] = 'spins around its orbit so both
                ↪  sides get heat from the sun'

        else:
            dataset.at[index, 'tidal_locked'] = 0.0

    return dataset

exoplanet_data = tidal_locking(exoplanet_data)
training_data = tidal_locking(training_data)
```

Figure 50: Getting Stellar and Planet Size as a Ratio

```
exoplanet_data['stellar_planet_ratio'] = exoplanet_data.apply(lambda row:
             ((row.st_mass / (row.pl_bmasse)) * 100) if row.st_mass !=0 and
             ↪  row.pl_bmasse !=0 else 0, axis=1)

training_data['stellar_planet_ratio'] = training_data.apply(lambda row:
            ((row.st_mass / (row.pl_bmasse)) * 100) if row.st_mass !=0 and row.pl_bmasse
            ↪  !=0 else 0, axis=1)
```

Figure 51: Getting Prompts

```
def get_prompts(dataset):
    for index, data in dataset.iterrows():
        #creating a prompt with star size and planet size as numbers (our foundation
        ↪  prompt)
        dataset.at[index, 'mass_prompt'] = "A solar system made up of {} planet(s), {}
        ↪  star(s), and {} moon(s). This {} planet is {} the size of earth, {} and {}.
        ↪  This planet {}. The planet\'s star is {} and {} the size of the sun.".format(
            data['sy_pnum'], data['sy_snum'], data['sy_mnum'], data['planet_category'],
            ↪  f"{pl_bmasse_int} times" if pl_bmasse_int != 0 else "an unknown size
            ↪  compared to", data['planet_color'], data['planet_spin'],
            ↪  f"{data['tidal_locked']}" if data['tidal_locked'] != 0 else 'has an
            ↪  unknown spin', data['stellar_color'], f"{st_mass_int} times" if
            ↪  st_mass_int != 0 else "an unknown size compared to")

        #creating a prompt with star size and planet size as a ratio
        dataset.at[index, 'ratio_prompt'] = "A solar system made up of {} planet(s), {}
        ↪  star(s), and {} moon(s). This {} planet is {} the size of earth, {} and {}.
        ↪  This planet {}. The planet\'s star is {} and {} the size of it\'s
        ↪  planet.".format(
            data['sy_pnum'], data['sy_snum'], data['sy_mnum'], data['planet_category'],
            ↪  f"{pl_bmasse_int} times" if pl_bmasse_int != 0 else "an unknown size
            ↪  compared to", data['planet_color'], data['planet_spin'],
            ↪  f"{data['tidal_locked']}" if data['tidal_locked'] != 0 else 'has an
            ↪  unknown spin', data['stellar_color'], data['stellar_planet_ratio'])

        #creating a prompt with star and planet size as text
        dataset.at[index, 'size_text_prompt'] = "A solar system made up of {} planet(s),
        ↪  {} star(s), and {} moon(s). This {} {} planet {}. This planet {}. The
        ↪  planet\'s star is {} and {}.".format(
            data['sy_pnum'], data['sy_snum'], data['sy_mnum'],
            ↪  data['planet_mass_description'], data['planet_category'],
            ↪  data['planet_color'], data['planet_spin'], f"{data['tidal_locked']}" if
            ↪  data['tidal_locked'] != 0 else 'has an unknown spin',
            ↪  data['stellar_color'], data['stellar_mass_description'])

        #creating a prompt with 75 tokens
        dataset.at[index, '75_tokens'] = 'A {}, {} star with a {}, {} planet. The planet
        ↪  {}, {}, and {}'.format(data['stellar_color'],
        ↪  data['stellar_mass_description'], data['planet_mass_description'],
        ↪  data['planet_category'], data['planet_color_short'],
        ↪  data['planet_spin_short'], data['tidal_locked'])

    return dataset
```

Figure 52: Getting Metadata File

```
import pandas as pd
import json

metadata = {}

for index, data in dataset.iterrows():
    image_path = data['image_path'].split("/")[1].split(".")[0]
    image_data = {
        "tags": "solo, no humans, space, starry night",
        "caption": data["mass_prompt"]
    }

    metadata_long[image_path] = image_data

output_file = "metadata.json"

with open(output_file, "w") as json_file:
    json.dump(metadata_text, json_file, indent=4)
```

14 References

Automatic (2022) [Bug]: *Training embedding with 2.0 works, 2.1 returns "nan"*· Issue #5715 · AUTOMATIC1111/stable-diffusion-webui.Availableat:https://github.com/AUTOMATIC1111/stable-diffusion-webui/issues/5715.

Baraffe, I. et al. (2003) *"Evolutionary models for cool brown dwarfs and extrasolar giant planets. The case of HD 209458,"* Astronomy and Astrophysics, 402(2), pp. 701–712. Available at: https://doi.org/10.1051/0004-6361:20030252.

Beaty, M (2023) *Thesis Repository.* Github. Available at: https://github.com/mbeaty2/thesis

CalTech (2014) *Getting To Know Super-Earths.* Available at: https://www.caltech.edu/about/news/getting-know-super-earths-44005.

CalTech (2021) *Planetary Systems and Planetary Systems Composite Parameters Data Column Definitions.* Available at: https://exoplanetarchive.ipac.caltech.edu/docs/API_PS_columns.html.

Cranmer, S.R. (2021) *"Brown Dwarfs are Violet: A New Calculation of Human-eye Colors of Main-sequence Stars and Substellar Objects,"* Research Notes of the AAS, 5(9), p. 201. Available at: https://doi.org/10.3847/2515-5172/ac225c.

Cuenca P., Paul S. and HuggingFace (2023) *Using LORA for efficient stable diffusion Fine-Tuning.* Available at: https://huggingface.co/blog/lora.

Dosopoulou, F., Naoz, S. and Kalogera, V. (2017) *"Roche-lobe overflow in eccentric Planet–Star systems,"* The Astrophysical Journal, 844(1), p. 12. Available at: https://doi.org/10.3847/1538-4357/aa7a05.

Duan, B. et al. (2019) *"Correlation of absorption spectrum properties of methane with coupling variables between temperature and concentration,"* Optik, 185, pp. 82–87. Available at: https://doi.org/10.1016/j.ijleo.2019.03.070.

Educative (2020) *How to use an API: fetch daily images with NASA's Open API.* Available at: https://www.educative.io/blog/how-to-use-api-nasa-daily-image.

ESA (2012) *Chasing clouds on Venus.* Available at: https://www.esa.int/Science_Exploration/Space_Science/Chasing_clouds_on_Venus#:~:text=Clouds%20regularly%20punctuate%20Earth's%20blue,dioxide%20and%20sulphuric%20dioxide%20haze.

Geballe, T.R. et al. (2002) *"Toward spectral classification of L and T dwarfs: Infrared and optical spectroscopy and analysis,"* The Astrophysical Journal, 564(1), pp. 466–481. Available at: https://doi.org/10.1086/324078.

Guzewich S., Lustig-Yaeger J., Evan Davis C., Kumar Kopparapu R., Way M., Meadows S. (no date) *"The Impact of Planetary Rotation Rate on the Reflectance and Thermal Emission Spectrum of Terrestrial Exoplanets Around Sun-like Stars,"* NASA Goddard Space Flight Center [Preprint]. Available at: https://ntrs.nasa.gov/api/citations/20205001155/downloads/20-20.pdf.

Habets, G. M. H. J.; Heinze, J. R. W. (1981) *"Empirical bolometric corrections for the main-sequence"*. Astronomy and Astrophysics Supplement Series: A European Journal. [PDF]. 46th edn (1997).

Harvard (no date) *Spectral classification.* Available at: https://lweb.cfa.harvard.edu/~pberlind/atlas/htmls/note.html.

HuggingFace (no date) *Low-Rank Adaptation of Large Language Models (LORA)*. Available at: https://huggingface.co/docs/diffusers/training/lora.
JPL-Caltech (2018) *Space Tourism Posters — NASA Solar System Exploration*. Available at: https://solarsystem.nasa.gov/resources/682/space-tourism-posters/.
Kohya (2023) *Training Stable Diffusion*. Google Colaboratory. Available at: https://colab.research.google.com/drive/1ZVukUuUMLxIZ6BgX7loKSMxcoBhfg70B#scrollTo=XhXhQY5Sov-g.
Li, J. et al. (2022) *BLIP*. Available at: https://huggingface.co/docs/transformers/model_doc/blip.
NASA (2015) *NASA Exoplanet Archive*. Available at: https://exoplanetarchive.ipac.caltech.edu/index.html.
NASA (2016) *Planetary fact sheets*. Available at: https://nssdc.gsfc.nasa.gov/planetary/planetfact.html.
NASA (2021) *Overview — What is an Exoplanet? – Exoplanet Exploration: Planets Beyond our Solar System*. Available at: https://exoplanets.nasa.gov/what-is-an-exoplanet/overview/.
NASA (2022) *Goldilocks Zone - Exoplanet Exploration: Planets Beyond our Solar System*. Available at: https://exoplanets.nasa.gov/resources/323/goldilocks-zone/.
NASA (2022) *Spectroscopy: Reading the rainbow*. Available at: https://hubblesite.org/contents/articles/spectroscopy-reading-the-rainbow.
NASA (2022) *Overview — Planet Types – Exoplanet Exploration: Planets Beyond our Solar System*. Available at: https://exoplanets.nasa.gov/what-is-an-exoplanet/planet-types/overview/.
NASA (2022) *Solar System Temperatures — NASA Solar System Exploration*. Available at: https://solarsystem.nasa.gov/resources/681/solar-system-temperatures/#:~:text=Planetary%20surface%20temperatures%20tend%20to,%C2%B0F%20(167%C2%B0C).
NASA (2022) *Stars — What is an Exoplanet? – Exoplanet Exploration: Planets Beyond our Solar System*. Available at: https://exoplanets.nasa.gov/what-is-an-exoplanet/stars/.
NASA (2023) *images.nasa.gov API Documentation*. Available at: https://images.nasa.gov/docs/images.nasa.gov_api_docs.pdf(Accessed:August21,2023).
NASA (no date) *Basics of Space Flight - Solar System Exploration: NASA Science*. Available at: https://solarsystem.nasa.gov/basics/solartemperature/.
Natsume, Y. (2023) *"Using Long Prompts with the Diffusers Package with Prompt Embeddings,"* Medium, 28 July. Available at: https://medium.com/mlearning-ai/using-long-prompts-with-the-diffusers-package-with-prompt-embeddings-819657943050.
Navascués, D.B. (2020) *Planetary Systems and the Habitable Zone*. OpenMind. Available at: https://www.bbvaopenmind.com/en/science/physics/planetary-systems-and-the-habitable-zone/.
NIST (2023) *Chemical name search*. National Institute of Standards and Technology. Available at: https://www.nist.gov
O'Connor, R. (2023) *"Stable Diffusion 1 vs 2 - What you need to know,"* News, Tutorials, AI Research [Preprint]. Available at: https://www.assemblyai.com/blog/stable-diffusion-1-vs-2-what-you-need-to-know/.

OpenAI Platform. *Tokensizer*. Available at: https://platform.openai.com/tokenizer.

OpenStax (no date) *7.2 Composition and structure of planets*. University of Central Florida. Available at: https://pressbooks.online.ucf.edu/astronomybc/chapter/7-2-composition-and-structure-of-planets/#:~:text=The%20Terrestrial%20Planets&text=In%20addition%20to%20being%20much,most%20common%20metal%20is%20iron.

Society, P. (2020) *The different kinds of exoplanets you meet in the Milky Way*. Available at: https://www.planetary.org/articles/the-different-kinds-of-exoplanets-you-meet-in-the-milky-way.

Stable Diffusion (no date) *Low-Rank Adaptation of Large Language Models (LORA)*. Available at: https://huggingface.co/docs/diffusers/training/lora.

Strubell, E., Ganesh, A. and McCallum, A. (2019) *"Energy and Policy Considerations for Deep Learning in NLP,"* Association for Computational Linguistics, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Available at: https://doi.org/10.18653/v1/p19-1355.

Sudarsky, D., Burrows, A. and Pinto, P.A. (2000) *"Albedo and reflection spectra of extrasolar giant planets,"* The Astrophysical Journal, 538(2), pp. 885–903. Available at: https://doi.org/10.1086/309160.

Team, K. (2023) *Keras documentation: Fine-tuning Stable Diffusion*. Available at: https://keras.io/examples/generative/finetune_stable_diffusion/.

The Last Ben (2023) *Fast-Stable-Diffusion*. Google Colaboratory. Available at: https://colab.research.google.com/github/TheLastBen/fast-stable-diffusion/blob/main/fast-DreamBooth.ipynb#scrollTo=Baw78R-w4T2j.

Tryolabs (2022) *The guide to fine-tuning Stable Diffusion with your own images*. Available at: https://tryolabs.com/blog/2022/10/25/the-guide-to-fine-tuning-stable-diffusion-with-your-own-images.

University of Oregon (no date) *Uranus/Neptune*. Available at: https://pages.uoregon.edu/jschombe/ast121/lectures/lec20.html.

Vanderbilt University (no date) *Kepler's 3rd Law*. Available at: https://www.vanderbilt.edu/AnS/physics/astrocourses/ast201/keplerslaws_3.html.

Vincent, J. (2023) *"Getty Images sues AI art generator Stable Diffusion in the US for copyright infringement,"* The Verge, 6 February. Available at: https://www.theverge.com/2023/2/6/23587393/ai-art-copyright-lawsuit-getty-images-stable-diffusion.

Wiki, C. to T. (no date) *"Gas Giants (Theoretical Models),"* Terraforming Wiki [Preprint]. Available at: https://terraforming.fandom.com/wiki/Gas_Giants_(Theoretical_Models).