i## **Concept Statement**

Our team decided to use the natural beauty of space as an inspiration to generate art with the same wide variety of structure and color, and use the range of color present in photos of space to color photos in different contexts.

Github Link

https://github.com/rohany/artml-final-project

Process

Since our project aimed to build on top of our work in Project 2, we started where we left off there, with an image dataset of ~1000 images, and some preliminary experimentation with thresholding and grayscale preprocessing methods.

The first hurdle we ran into was getting the Pix2PixHD network to run on our original dataset. The network had numerous undocumented requirements for the input, and took time to identify these various requirements. For example, the network required all input images to have a height and width that is divisible by 32. If this requirement was not met, the network failed with an arbitrary shape mismatch during training.

After we were able to get the network running, we began using our existing pipeline and data to train the generator using Pix2PixHD. We ran experiments with both of our preprocessing methods, and found unsatisfactory results. The images we generated were still quite blurry and low quality, and did not compare well to the target image. After more investigation, we realized that this was due to problems in our original dataset. Our original dataset was created manually, and had a large amount of noise within it. The original dataset was full of images of varying size and quality, and many images had to be rescaled, shrunk, or stretched before being fed into the network. All of these factors led to a poor training set, and unsatisfactory results from our network.

To address the issues within our dataset, we decided to remake our dataset, but this time include only high quality images of the correct size, to avoid distortion in training. There are relatively few high quality images of space and celestial structures, but most of them are available through the hubble space telescope website. To collect our new dataset, we wrote a web scraping script to follow the thumbnail links in the website and automatically download the 1024x1024 version of the image that each thumbnail pointed to. Using this method, we were able to collect nearly 2100 high quality, square images of space and galaxies. Lastly, we manually inspected the dataset to remove images that had words, graphs, or labels on them. These extra elements in the image would only lead to more noise in training. After this process, we had collected around 2000 images to form a dataset that was high quality enough to use to train our network.

To correctly use our new dataset, we had to adjust some parameters used in our thresholding methods. Additionally, we added an edge detection preprocessing stage to our set of preprocessing steps, which is something that we wanted to try for Project 2, but didn't get a chance to. Our hope was that the edge detection might make a better representation of what a human would draw, compared to the thresholding. The edge detection might also give more information about the shape of structures in space to the generator network.

After adjusting our preprocessing and adding edge detection as a preprocessing method, we began to train Pix2PixHD on our 3 datasets -- thresholded, gray scale, and edges. Our target images have dimensions 768x768. We wanted to train to generate 1024x1024 images, but this required at least 16 GB of GPU memory. Since these networks take a much longer time to train at higher resolutions, we decided it was not economical to use a p2.8xlarge or a p3.xlarge instance to have access to 16 GB of GPU memory. Training a 768x768 network only needed around 9 GB of GPU memory, making the p2.xlarge instance a good choice. We trained our networks for 5-6 days each. We trained the edges network only on 512x512 to make sure that we had at least 1 network complete a larger amount of iterations by the time the project deadline came around.

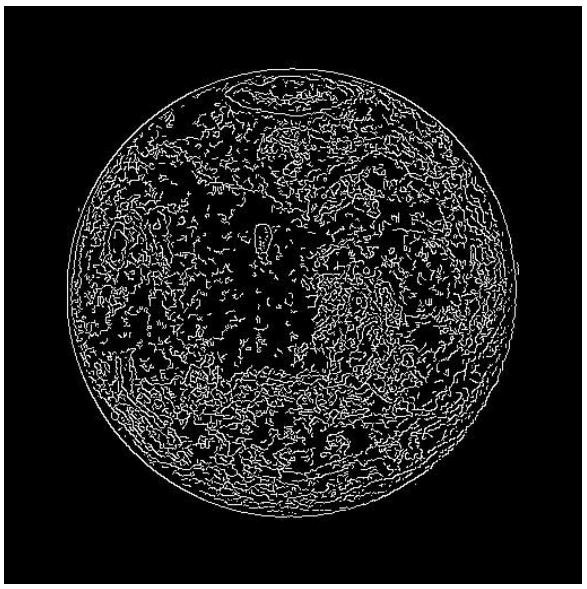
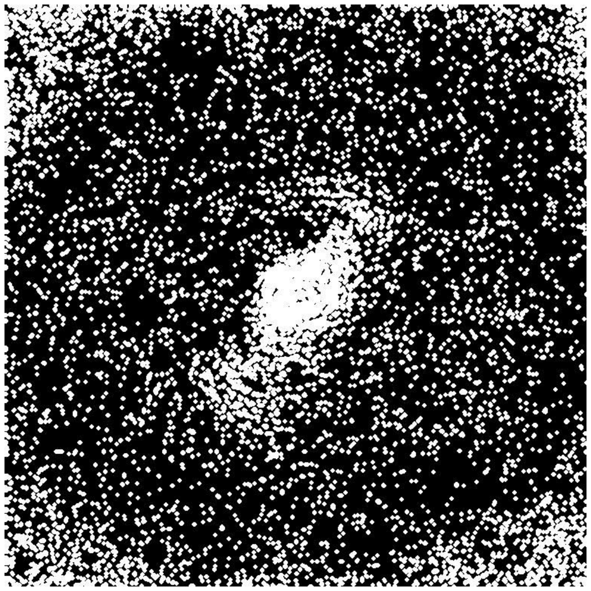
During training the networks, we were able to monitor their progress, and see what kind of images they are generating at different stages of the training.

The first network to discuss is the grayscale network. We saw very promising results from using grayscale preprocessing in Project 2, letting us color grayscale photos that weren't galaxies with a similar vivid color scheme. However, after many days of training with Pix2PixHD, we saw that our network was not even close to as successful as it was in Project 2. The network was able to learn the shape and structure of the target image very well, but was not able to learn the color well at all, as seen in our results section. In addition, the images were not as high resolution as the originals and were in fact slightly grainy. Notably, after the first few iterations it seemed that the network was not improving at all. The network was able to learn only a specific dull, yellowish color scheme, in stark contrast to the vibrant colors from Project 2. On a large number of input images, the network barely colors the image at all, and the result looks grayscale itself. We hypothesize that the network ended up mainly learning mapping from particular grayscale values to a color, and was not able to use the shapes and structures within the input images to infer more interesting uses of color. Additionally, it is possible that our network in Project 2 was able to perform well only due to the low resolution of the target image. Once scaling to higher resolutions, the network was not able to learn the same things it did at lower resolutions. To see if higher resolution was the main issue, we additionally tried training the network on grayscale images but at the 512x512 resolution. Even at the lower resolution we saw similar poor results, which surprised us again. Another possible explanation is that the network quickly reached a local minima. With the variety of colors and the simplistic, direct mapping from grayscale to color, perhaps the network decided upon an average or common color from the pictures in order to minimize error. It is possible that the mapping from grayscale to colored is too direct to induce creativity. Perhaps some variance in both color and shape is needed. Based on these results, we decided not to include the results from this network in our final exhibition, but have provided some samples for the purpose of this report. We instead include results from our other network which was much more successful.

Next, we trained on the data preprocessed with an adaptive thresholding algorithm. The adaptive thresholding applies a threshold using a value based on the average pixel value within a small window. This allows for local differences in color and intensity to be represented in the final thresholded output. In contrast to the network trained on grayscale images, our network trained on thresholded images performed very well, in some cases generating images that looked better than the source image. The network is sometimes not able to learn some of the intricate patterns and shapes that appear in the source images, but overall generates good results. There are some interesting observations that arise from the preprocessing. The network does not generate quality images when there is a large amount of black or white space in the thresholded image. In this case, the generated images are very fuzzy and unsatisfying. The consequence of this is that the type of input images that create quality output from the network might not be created that easily by a human.

Lastly, we trained on data that has been preprocessed and replaced by the edges within original image. We used canny edge detection algorithm provided by OpenCV to implement this transformation. We trained our network at the resolution of 512x512 to have at least one network that would complete training a large number of epochs by the time our AWS budget ran out. The network performed extremely well on training, and was able to almost directly create the training image from the edges in the input label. While these results were promising, the processed images from the edge detection algorithm seem difficult to draw by hand. In general, the preprocessing from the edge detection included a high amount of detail in the input image to the network, which allowed the network to perform very well.

Examples of preprocessed input from both the thresholding and edge detection preprocessing:

To generate our final results, we selected a set of generated images during training and evaluating our networks, as well as art generated from the network applied to hand made drawings. We could not really generate good artwork using purely hand drawings, as it is difficult to replicate the detail in the preprocessed inputs. Therefore, we used different brushes and filters in photoshop to generate relatively detailed black and white images, and let the network fill in colors and shapes from our drawings.

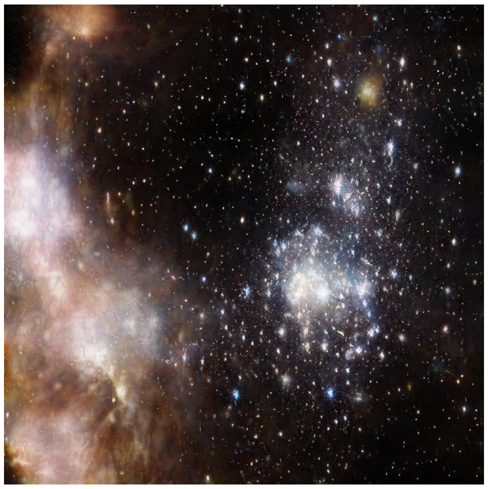
Documentation of trials and errors

- Rohan's Documentation
  - Worked to get Pix2PixHD network to successfully train on a dataset
  - Experimented with original dataset and preprocessing
  - Wrote web scraping script to collect new high definition dataset
  - Manually pruned new dataset to exclude noisy images
  - Extended preprocessing set to include edge detection
  - Setup infrastructure to allow easy training for group members
  - Spent 1 week training the network on the thresholded dataset
- Will's Documentation
  - Trained a network on 512x512 images for a week.
  - Hand drew some black and white art for the network to generate images from.
    - Network didn't do well with solid or thick blocks of white. It wanted fine and dense textures in order to generate good images.
    - Stippling patterns or dense linework worked best.
  - Designed and made posters for the exhibition
- Dennis's Documentation
  - Spent 1 week training the network on the grayscale dataset
  - Discovered the grayscale network's shortcomings and attempted to rectify

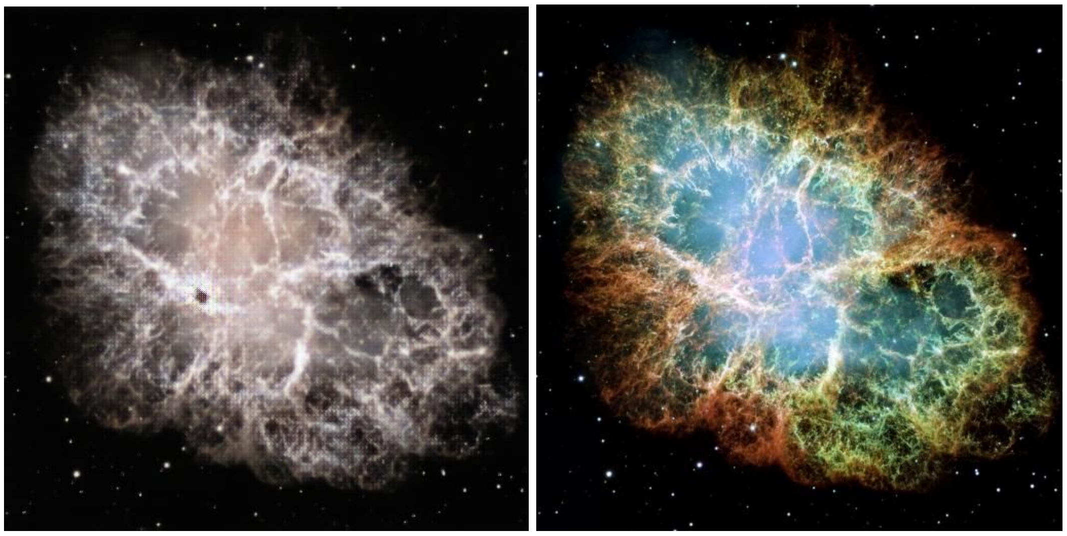
    - Modified images to 512x512 from 768x768 in order to attempt to get better results
    - Considered the reason behind the failure of the grayscale network
    - Picked some images from the grayscale network to display its faults
- Maggie’s documentation
    - Created a few black and white drawings for the network to generate coloured images

Results

Generated images during training from the thresholding network:

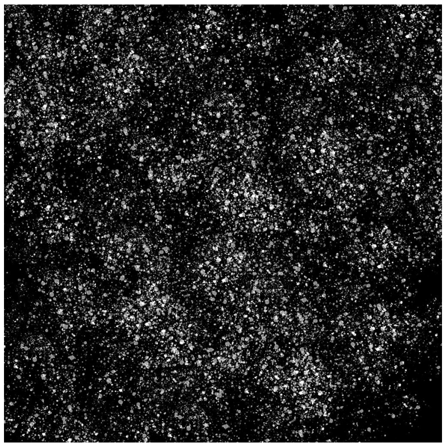
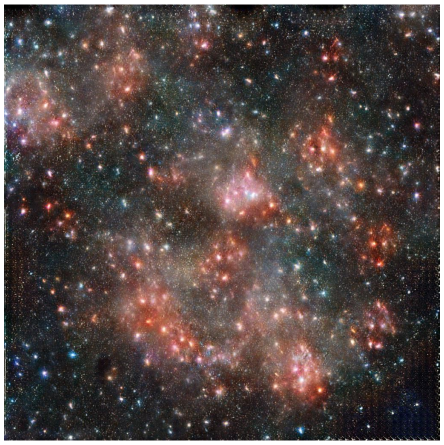
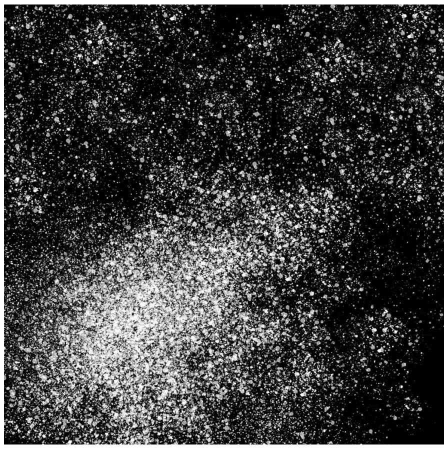
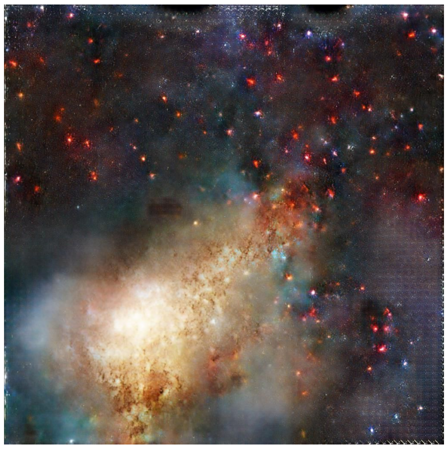


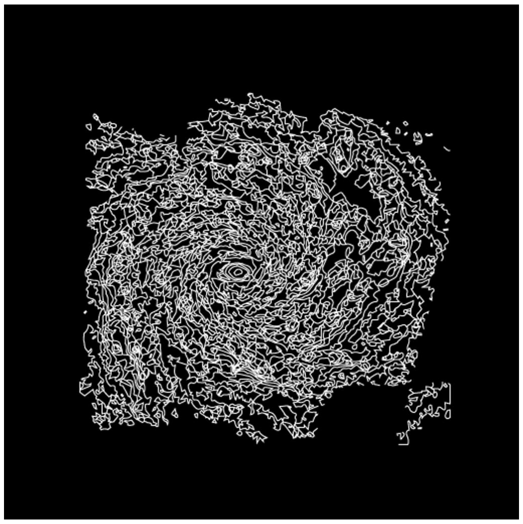
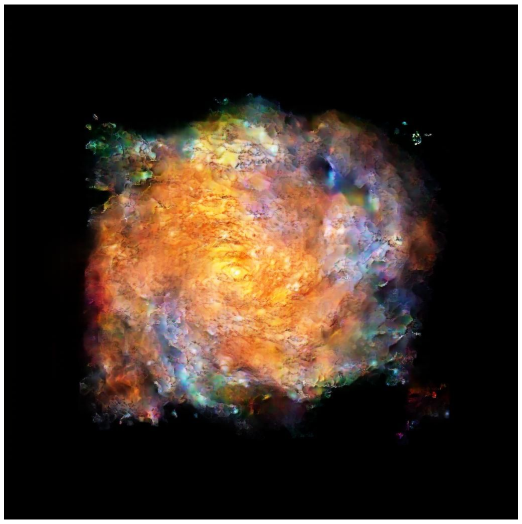
Generated images during training from the grayscale network (did not perform well). Original images on the left, generated images on the right:



Generated images during training from the edges network:

Generated images from hand drawn human input:

Self Evaluation of results

- Will’s self-evaluation
    - I liked the results of our continuation of project 2. We made some really cool stuff and got better results after taking more time to train the networks.
- Rohan’s self-evaluation
    - Overall I’m very happy with the results of this project. It was a good extension to our Project 2, and I think we generated some very high quality images.
- Dennis’s self-evaluation
    - I think that we formed a good basis during a Project 2 and made this a good natural extension of that project. We received better results, took more time training the networks and introduced better thresholding techniques. As a result, we created some images that are truly beautiful.
- Maggie’s documentation
    - The coloured images generated from hand-drawn black and white images were much better improvements from Project 2 and the results were realistic and beautiful.

Group Member Contributions

- We all equally contributed to the project. We pooled the results we were satisfied with and picked from the best ones.