UNet Optimizations for Low Earth Orbit Satellite Imagery on Road Segmentation

Benjamin Miller, Tyler Hilbert, Ruihong Lyu

***Abstract*—UNet can be used to segment specific components in images. Due to this, it is possible to use UNet on geographic satellite images to create road maps. Given the complex nature of having to learn how to segment out roads from satellite images, training this network can take a large amount of time. Due to this, being able to improve the performance of the UNet without having to perform a lot of extra training every time new types of road structures that have not been seen before are introduced is important. We thus sought to see if we could improve the performance of a readily available UNet structure available on GitHub [1] by changing the default UNet architecture as well as by applying different types of pre-processing techniques on images before the UNet predicted on them to see if they improve the performance of the UNet. Additionally to make up for the low IoU score of UNet for low resolution road segmentation, we created our own road segmentation method utilizing a combination of traditional filtering techniques.**

I. Introduction

Due to increasing urban development planning, traffic routing, and planning for natural disasters there has been an increase in interest towards road mapping [2]. The complexity of urban areas makes this task very difficult using traditional image processing. [2]. This is mostly because traditional image processing requires prior information such as hand-crafted features and descriptors [2]. As has been noted, this means that traditional image processing techniques are limited by the data that has been fed into them [2]. Standard techniques used to improve performance include modifying parameters based off structural characteristics that the intended topography has. [2].

Regardless, it has been noted that these techniques took too much time and effort to perform. Additionally standard traditional techniques used to extract features were found to be prone to errors. [2]. This lead to a push to use Convolutional Neural Networks (CNNs) to extract features [2]. CNNs are noted to be very efficient and can generalize well [2]. However, CNNs take a lot of time to train and are also prone to overfitting [2]. They can also suffer from having vanishing gradients due to the non-linear activation functions [2]. Another alternative technique used morphological processes to try and extract roads [2]. However, the performance of these techniques relied on the structure element that was used and the specific operation that was used [2]. More advanced methods involved using textural features in SVMs and then using Markov Random fields to segment the roads in an image [2]. These techniques were found to fail when performed in areas high contrast and wide road segments [2].

UNet was originally invented in 2015 to improve the ISBI cell tracking challenge [3]. UNet is an image segmentation network used to identify pixels of a specific class in an image [3]. It utilizes a descending series of encoders that are each tied to an ascending series of decoders. The encoder follows the typical architecture of CNNs [3]. It utilizes a pattern of two NxN convolution kernels with ReLU activation functions and a 2x2 max pooling operation with stride 2 for downsampling [3]. This is then followed by techniques that decode the network back to its original shape [3]. The current industry standard achieved not using a UNet is 83 percent accuracy, while UNet achieved a 92 percent accuracy for segmenting cells in an image. This caused further investigation into how this network could be used to help in more image segmenting problems. The general structure of a UNet is seen in figure 1 [3].

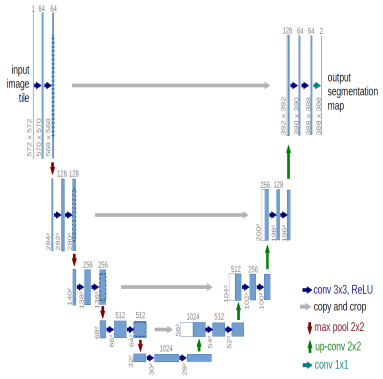

Fig. 1. General structure of the UNet

II. Project Description

A. Overview

Given the push for better road segmentation and the rising interest in the UNet structure, we wanted to see if we would be able to improve the performance of a readily available UNet architecture from GitHub [1] made to segment the roads in an image. We aimed to see if changing the structure of this network as well as other hyperparameters would improve its performance. We also sought to see if we could improve the performance of the network by applying pre-processing to images before they are fed into the network. The pre-processing was of notable interest as if pre-processing proved beneficial, then the performance of UNet could be increased without the need to retrain the network which is a very time consuming process. We worked off an example GitHub project mostly making modifications to the network and pre-processing [1].

Due to complications that will be addressed later, we also ended up adding more onto the project to make up for what we could not accomplish. Specifically, we created a road segmenting algorithm that attempted to create a mask corresponding to the location of the road using only satellite image processing techniques.

B. Responsibilities

Benjamin was responsible for creating all of the pre-processing filters as well as implementing the Marr-Hildreth filter. Benjamin also implemented the IoU scoring function as well as the road segmentation using standard image processing. Ruihong was responsible for creating the Canny-Edge detector as well as using the python notebook to shrink all of the images. Ruihong also performed training on the standard network as well as a smaller network to see the differences between the results. Ruihong also went through and isolated the 303 images we used to train a network to

produce some adequate results. Ruihong was also responsible for testing the pre-processed images as well as looking for differences between the produced masks. Tyler was responsible for training, testing, and altering the UNet to see whether changing the activation function, the number of layers, or the size of the layers could improve the performance of UNet. Tyler also wrote an program to rotate the images such that we could have more training data. Tyler also helped testing out multiple different boiler plate UNet projects to find one that was compatible with our environment and dataset. Tyler came up with the original project idea and identified the dataset used.

C. Training the UNet

To begin, we first needed to find a UNet structure and training code capable of learning to segment the roads in the image. These codes were found on GitHub using link in the following reference [1]. We have also tried to supply the version of the code that we modified to generate our final network topography as well. We also needed to find a relevant dataset that could be used to train the network. Most of our research into what dataset could be used pointed us toward the Massachusetts Roads dataset available at multiple sources though we downloaded ours from Kaggle [4].

Once we had the training code and the dataset, we utilized the supplied code to truncate the images to 256x256x3 from size 1500x1500x3. While this removes some precision, the size of the original data was too large to train on thus often caused the training process to crash. On top of this, running the custom pre-processing techniques on images of size 1500x1500 was very time consuming and was thus not practical given our timeline. We then attempted to train the network on these downsampled images. However, there were notable issues during the training process. Most of the time the network converged to creating all black masks, that is to say the network could not segment anything at all. We tried various different things such as doubling the size of each convolutional layer, changing the number of layers, modifying the activation function to tanh/ELU/SELU, changing the learning rate of the model, and modifying the batch size in the training process. For some reason, when changing batch size to one, UNet started to learn. This could be because the dataset size is small. We also attempted to improve the performance of the network by changing the loss function used during the training process to a dice coefficient, but this still did not produce great results. The end model trained converged on a local minimum of segmenting no roads in the images which was not an improvement over the original UNet.

D. Pre-processing filters

While there were various different filtering techniques that could all possibly improve the performance of the UNet, we decided to settle on the following filters:

- our implemented highpass filter
- our implemented lowpass filter
- our implementation of Laplacian of the image
- adding the Laplacian to the image
- our implementation of unsharpend filtering
- our implementation of highboost filtering
- a special filter we created using our implementation of the Marr-Hildreth edge detector
- a special filter we created using our implementation of the Canny-Edge detector

Since the road images were RGB images, we needed a way to filter over them. While it is possible to filter on all three color channels separately, we instead decided to create RGB to HSI and HSI to RGB functions. When applying our filters, we would first convert the image to its HSI version. We would then apply the filtering on the intensity image (not on Hue or Saturation). We then convert the image back to its RGB values post processing.

The highpass filter was set to allow frequencies of 5 and over. While originally we had a higher cutoff frequency which allowed for a majority of unimportant features to be eliminated (such as large patches of grass), we found at higher values parts of the road started to be eliminated from the image as. This lead to gaps in the road which we figured would make it harder for the network to process. Thus we eventually settled on a cutoff frequency of 5 as a balance between eliminating extraneous detail and keeping the road in tact. The lowpass filter had a cutoff frequency of 20. We also created a filter that calculated the Laplacian of the image. We theorized that the higher Laplacian values could potentially indicate the presence of the road, but the main reason we implemented the Laplacian was so that we could sharpen the image by adding the Laplacian to the image. We also decided to implement unsharp masking and highboost filtering to see if sharpening the images this way would make it easier for UNet to identify the roads. To clarify, unsharp masking involves blurring the original image and then subtracting the result from the original image to get a new one. This new image is then added to the original image by a factor of k. For unsharp masking we set k to be 0.75. When the value of k is above 1 this technique is called highboost filtering. For our highboost filter we set the value to be 1.25.

The final two filters we implemented were filter systems we created ourselves. Both of our special filters utilized a Gaussian filter that we implemented as well as the unsharp filtering system that we just described. The filter system also used either our Marr-Hildreth edege detection that we implemented or the Canny-Edge detection System that we implemented. The Marr-Hildreth edge dection system starts by first applying a Gaussian filter to the image. After this the Laplacian is calculated. The mask is then formed by finding the zero-crossing on this new image. We implemented the zero cross filter ourselves. This was performed by scanning through the Laplacian of the Gaussian and looking for where the sign changes. We also set a threshold where not only does the sign need to change, but the difference in values must exceed the threshold. We set the threshold to be 0.4. This then creates a mask. The canny edge detector is made by first applying a

spatial 5x5 Gaussian filter to the image. The Sobel kernel is then used to detect vertical and horizontal edge of the image. After this we implemented non-maximal suppression that is applied on the vertical, horizontal, and -45 and 45 degree angles. After this we implemented double thresholding and edge connectivity analysis to find the resulting mask. Based on our image data, a low threshold was set to 0.2 multiplies the mean of the column maximum of the suppressed gradient matrix and a high threshold is set to be twice the low threshold value to get the right amount of edge details. 8-connectivity is used to fill the gaps that could possibly exist along edges. To show that our implemented Marr-Hildreth and Canny-Edge detector both work we tested them on the moon.tif file that MATLAB has. Figures 2, 3, and 4 show the original image as well as the thresholds found by the Marr-Hildreth edge detector and the Canny-Edge detector. It was possible to change how much detail was kept in the moon by changing the thresholds. The two special filters do the following:

1) create a lowpass filtered version of the original image
2) create a unsharpended filtered version of the original image
3) use either the Marr-Hildreth edge detector or Canny Edge detector on the original image to create a mask.
4) Create a new image and go through every pixel performing the following:
   - if the mask has a 0: use the value from lowpass filter.
   - if the mask has a 1: use the value from the unsharpened image

Fig. 2. original image

E. Road detection using traditional filtering

We originally did not intend to perform road segmentation using traditional image processing techniques and did not mention this in our project proposal. However as we will expand on later, our UNet did not end up training as well as the industry standard. This meant that we were not able to go in depth on whether our techniques improved the UNet. Because of this, we decided to add onto our project by creating a road segmentation algorithm that utilizes traditional image filtering techniques. Our road segmentation algorithm to create the mask was as follows:

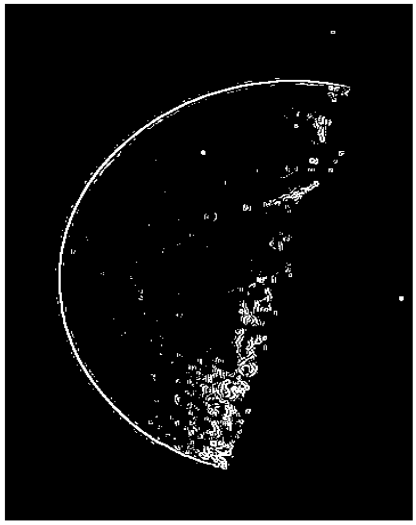

Fig. 3. edge found by Marr-Hildreth image

Fig. 4. Edge found by Canny-Edge image

1) calculate the HSI version of the image
2) make a new empty image
3) go through every pixel in the image and if the saturation value for that image was less than 0.08, set the RGB value of the new image to be equal to the RGB value of the original image.
4) convert the new image to its HSI equivalent
5) perform a dilation using the structure element [1 1 1; 1

1 1; 1 1 1] using our implemented dilation function.
6) perform connected component analysis starting on a part of the road using our implemented connected component function
7) using MATLABs bwmorph function, perform thinning.

We chose to keep pixel values with a saturation value of 0.08 as we noticed that most of the time the roads saturation value was below 0.08. The dilation was formed to close gaps caused by the previous step. The connected component analysis removed any random specs/dots on the image not connected to the road. The thinning was done to remove a lot of the random bulk picked up in the connected component analysis.

III. Results and Discussion

A. *Filtering results*

Figures 5 though 13 show examples of an original image as well as what the different filter techniques did.

Fig. 5. original image

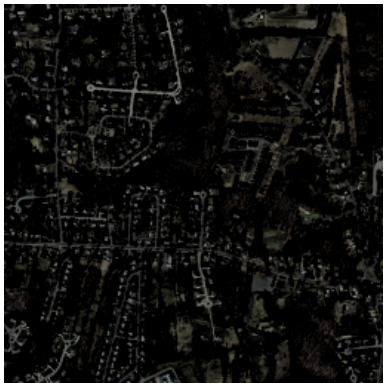

Fig. 6. highpassed image

Fig. 7. lowpassed image

Fig. 8. laplace image

Fig. 9. adding laplace image

Fig. 10. unsharped image

Fig. 11. highboost image

Fig. 12. Marr-Hildreth based special filter image

Fig. 13. Canny-Edge based special filter image

B. UNet

Our trained UNet did not do as well as the industry standard. It got stuck on a local minimum where it predicted all pixels were not road due to the spare nature of the road pixels. We tried various different modifications to the network including what was described earlier but most of the time the UNet produced a pure black image. Figure 14 shows an example.

Fig. 14. All Black example

We have a couple of possible explanations for this. One possible explanation for this is that parts of the dataset seem to be made for addressing the occlusion problem. In the case of occlusion, parts of the image are whited out but the ground truth mask is not. An example can be seen in figure 15. In this

Fig. 15. cropped image

case the network is trained to attempt to be able to predict roads when objects occlude them. However, it is possible that this interfered with the networks ability to learn. In an attempt to mitigate this effect, we went through and found 303 image that were not cropped and trained the UNet on this. Once again, these could not produce good results most of the time considering the small data size. However, on some of the training images it could produce some decent results, but failed to produce anything of value on test images. This seems to indicate that the network was overfitting the data, and even then it still only performed decently well for some of the training images. What this meant was that our original plan to test the filtered images on the trained network and compare the results could not be performed. However, we were able to at least somewhat test the results of filtering. For example, on one of the images that the UNet tried to predict on, it seems like the low pass filter may have been able to increase the performance as seen by figure 16,17, and 18.

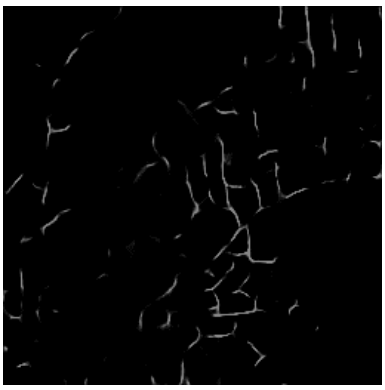

Fig. 16. mask predicted on original image image

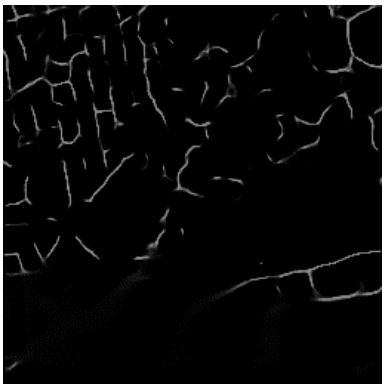

Fig. 17. mask predicted on lowpass image image

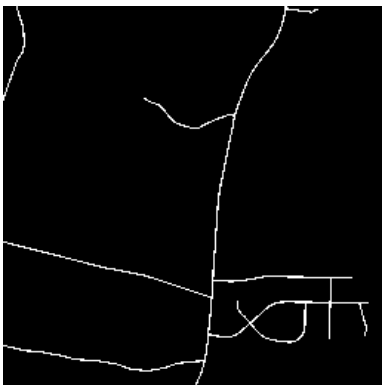

Fig. 18. original mask image

As these figures show, the lowpassed version of the image helped the broken UNet create part of the complex structure at the bottom of the image it was not able to create when predicting on the original image. Figure 19 is the ground truth mask. To see if we could get any more evidence that

the UNet's performance was being improved by the pre-process filtering, we ran a test image through the different pre-processing techniques and looked at the predicted masks. Figures 19-28 are the results of applying our filters in an image before trying to predict on them as well as the ground truth mask of the image.

Fig. 19. ground truth mask

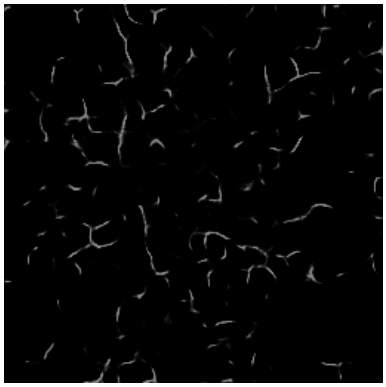

Fig. 20. original image

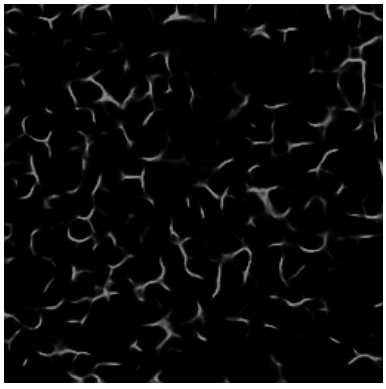

Fig. 21. highpassed image

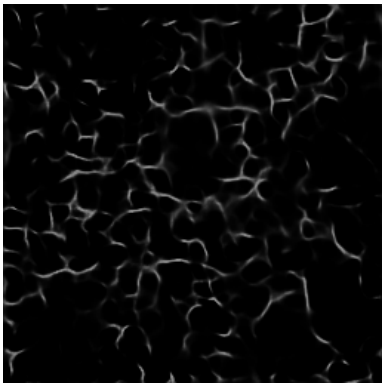

Fig. 22. lowpassed image

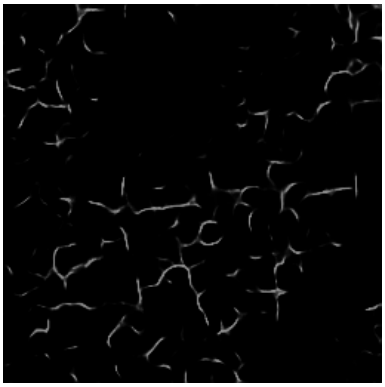

Fig. 23. lapplace image

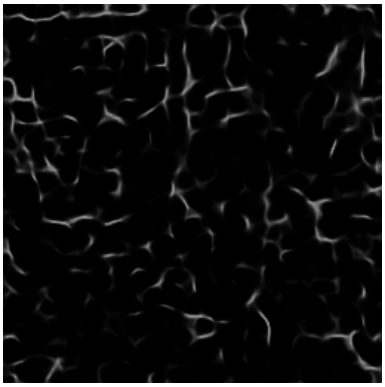

Fig. 24. adding laplace image

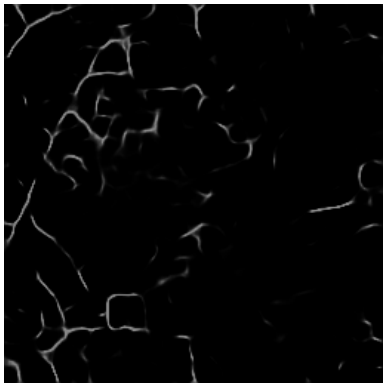

Fig. 25. unsharped image

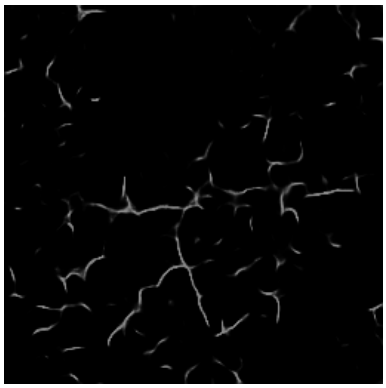

Fig. 26. highboost image

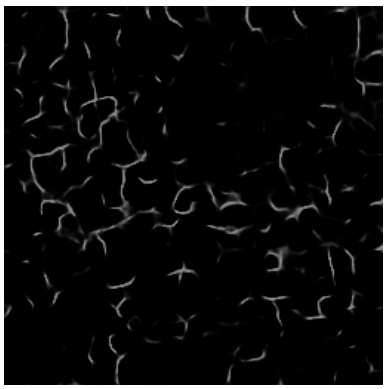

Fig. 27. Marr-Hildreth image

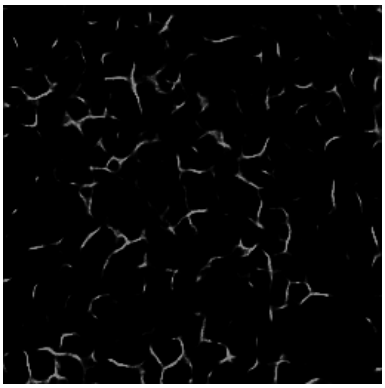

Fig. 28. Canny-Edge image

In this case looking at the original mask (Figure 18) and the predicted mask from the unsharp filter (Figure 24), it can be seen that the unsharped mask did outperform the original mask. To further check this,we implemented an algorithm to calculate the IoU score. The IoU score is defined by the following equation:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{1}$$

[5] and got the following results: The original image predicted mask had a score of 0.0017. The unsharped image predicted mask had a score of 0.0142. This means that the unsharped image performed nearly ten times as well as the original image. Again while no conclusive results can be obtained due to the UNet not training as well as the industry standard, we do have some results, mainly the low pass mask from figure 17 and the unsharped mask from figure 25 that suggest that using filters may be able to increase the results for the UNet.

While we were mainly focused on trying to improve the UNet performance, we were also curious about how the UNet learned. One of the ways we investigated this was by saving sets of masks that the UNet created after multiple 30 epochs of training. While most of the final masks were poor, we were able to find a few masks that looked good after 90 epochs. Since we saved what the model predicted between every 30 epochs, we investigated what the earlier versions of the model predicted for a single image. These can be seen in figures 29, 30, and 31. Based off these images, the UNet will first learn a general cloudy layout and slowly over time condense this cloudy region into more road like structures. This is interesting as we assumed the UNet would start by producing bulky lines and slowly refine them into smaller lines as time went on.

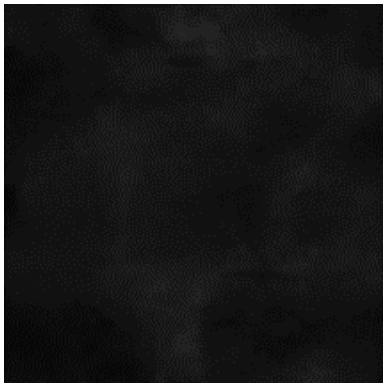

Fig. 29. results after 30 epochs

We also performed this procedure on a larger UNet model. However we found that the results were significantly worse for this increased network so instead checked what it produced after 180 and 280 epochs. The results can be seen in figures 32,33, and 34.

As can be seen in these images, the larger network was not able to learn as well as the smaller network even when it had more time. We initially had assumed that the more complex network would have an easier time learning, but this was not the case. Even more interesting, the larger network was never able to make the cloudy regions that the smaller network did before it learned how to create the roads. This makes us think the production of the foggy areas early in the training process has a major factor on whether or not the network will be able to produce decent images later on.

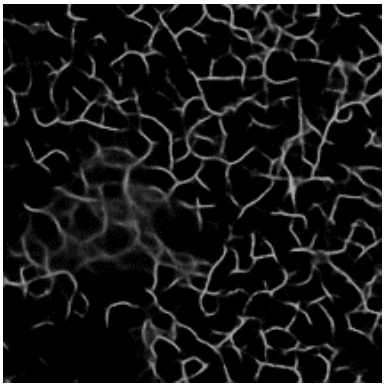

Fig. 30. results after 60 epochs

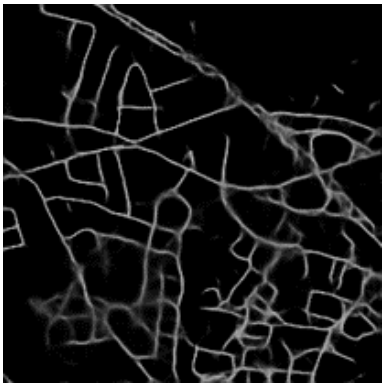

Fig. 31. results after 90 epochs

Fig. 32. results after 10 epochs

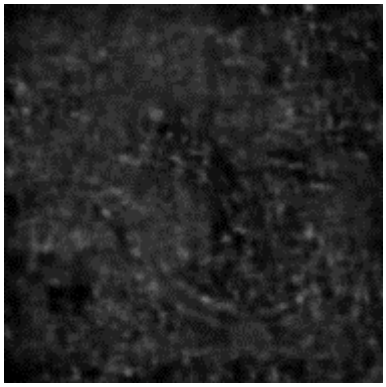

Fig. 33. results after 180 epochs

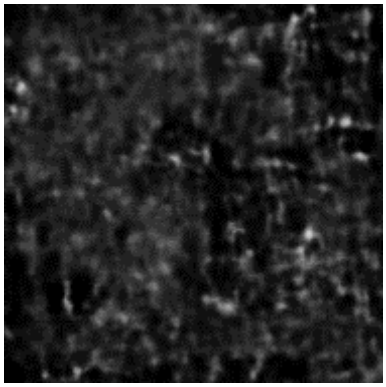

Fig. 34. results after 280 epochs

C. Traditional techniques image processing

The final thing we did for our project was the road segmentation described earlier using classical filtering techniques. To show the results of our algorithm we decided to perform our algorithm on one of the images. The results of each step of the process are shown in figures 35-38.

Fig. 35. Original image

While this algorithm was able to extract the general outline of the image, it is not able to match the performance of a working UNet model. This highlights the need for better road segmentation techniques that led to people leaving traditional techniques and towards the UNet. It also must be known that part of our algorithm utilized connected component analysis. This means that some prior knowledge of where part of the road is was required. We specifically only used one starting location, but the fact that a starting location needs to be known means that this algorithm needs some prior information to be run. Something worth mentioning is the result after applying the saturation filter, as the image that it produced was also a decent looking map.

Fig. 36. Saturation filtered

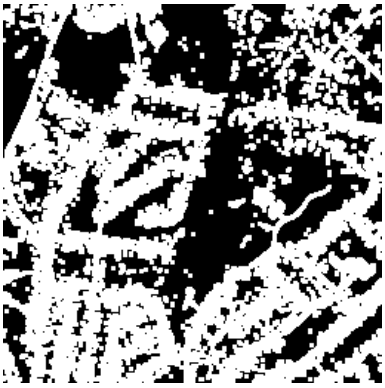

Fig. 37. Dilated

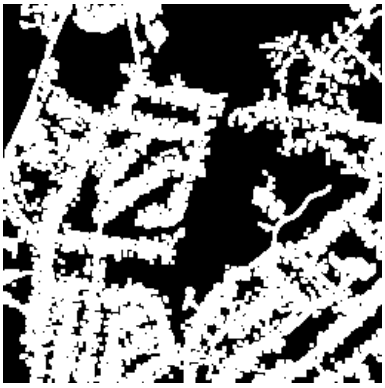

Fig. 38. Connected Component

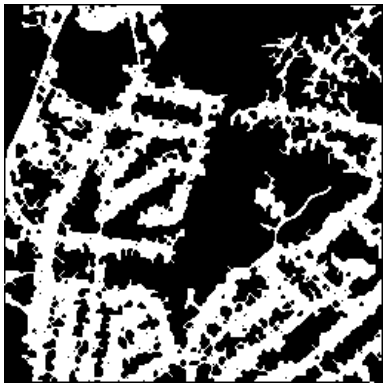

Fig. 39. Thinned

IV. Future Work

The largest future work item is to continue testing modifications to UNet to see if any of them can out perform the industry standard. Additionally, further testing to verify if the filtered images produced better masks when ran through the trained network would need to be done. There is some evidence that warrants further exploration into this topic of filtering before classification.

V. Summary and Conclusion

In this project we sought to find a way to improve the performance of an already built UNet training system found on GitHub [1] and modified the structure of the base UNet to see if we could improve its performance. The ways in which we sought to improve the performance of the network included modifying the architecture of the network as well as applying different filter techniques to the images before they are tested on the Unet. The modifications we tested did not result in a better model than the industry standard. This prevented us from having a clear conclusion on whether or not the filters improved the performance of the network. However, our results on the poor performing UNet seem to indicate that there is potential for the different filters to improve the performance of the network. This was more readily seen on the lowpass filter result on figure 17 and the unsharp filter result for figure 25 which warrants further investigation into the topic.

We also were able to create a road segmentation technique using traditional filtering techniques that produces a decent mapping of the roads on this dataset. What we learned from this is that there is a limit to how good traditional image filtering techniques can be due to needing some prior information about the image. In our case our system needed to know the saturation levels of the road as well as one pixel location that the road is on for connected component analysis. A UNet system does not need this and these limitations help explain why UNet is becoming a much more common technique.

References

[1] F. Maqbool, "Roads-segmentation-mnih-dataset (satellite images)." https://github.com/faisalmaqbool94/Roads-Segmentation-Mnih-Dataset/, 2019.

[2] M. S. Dey, U. Chaudhuri, B. Banerjee, and A. Bhattacharya, "Road segmentation using u-net architecture," 2022.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[4] V. Mnih, "Machine learning for aerial image labeling." https://www.kaggle.com/datasets/balraj98/massachusetts-roads-dataset/, 2013.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," vol. 9351, 2015.