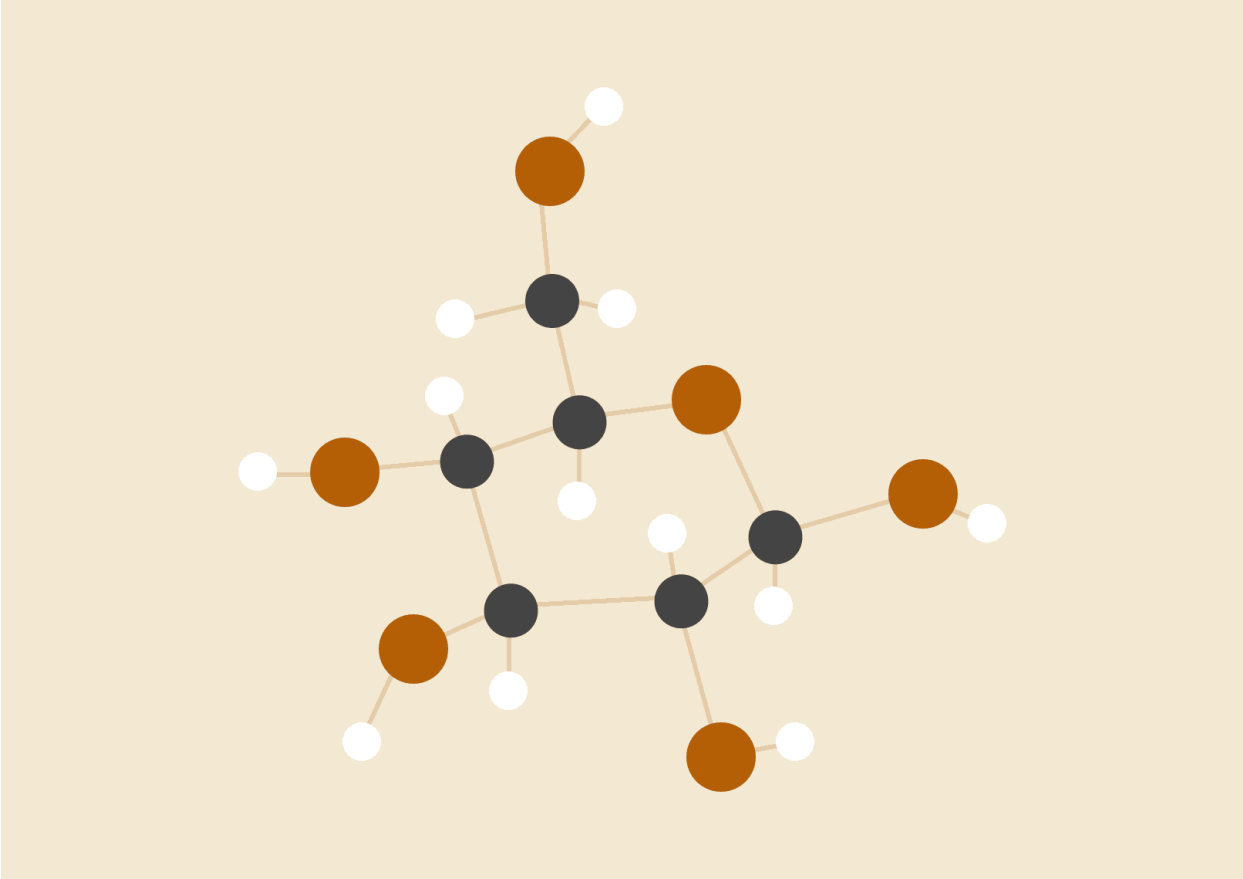
IMAGE SEGMENTATION USING NEURAL NETWORK

Under the guidance of Prof. ChenChen Liu



By Smith Christian and Dhruv Patel

Project report

MOTIVATION

Nowadays, Image segmentation has become one of the major aspects to classify the content of the images in the computer vision field. Image segmentation is being used in various fields such as medical, autonomous driving vehicles, astronomical image classification, satellite image processing and many more. As the satellite image storage is being flooded every day by the geographical images of the Earth, Deep Learning technique offers great help dealing with the big dataset. However, the models require a gigantic amount of storage and computation time. We propose our models whose results are similar to the existing models but operated on limited storage and drastically reduces the training time of the model by a factor of 10. In this paper we propose our own custom build model which is named as “Double Unet” which is compared to our unet and keras Unet model. The Double Unet drastically reduces the training time which is taken by Unet developed by keras by almost a factor of 10, without significantly affecting the accuracy.

PROPOSED SOLUTION

Smith and Dhruv both work on data preprocessing where we analyze the dataset thoroughly to know the outliers in the image dataset which might bring down the accuracy of the neural network. Smith and Dhruv worked on the dataset to reduce its sparsity for which they reduced the dimension of the image to 256 and convert all the image to grayscale. Smith heavily worked on designing a unet from scratch and implemented Unet by keras these models will help us understand on designing our own model for implementation. Dhruv Extensively worked on designing a custom model which can mimic the existing Unet model but which runs much faster and occupies less space, which he named as Double Unet. Smith and Dhruv thoroughly compared all the implemented models and found that the Double Unet training time was reduced by a factor of 10 where the drop on accuracy was only 10 percent which was quite acceptable based on the size of the images.

PROJECT MECHANISM

Unet Model: We have used an A repeated application of Two 3x3 convolution which is followed by ReLU as an activation function. Furthermore, it is followed by a max pooling of 2x2 layers.. There are a total of blocks which are added to the Unet. The blocks have the following filters [64 128 256 512 256 128 64]. There are total params: 10,663,873, which are getting trained when compared to the Unet which pretrained resnet which has around 55,000,000 total parameters. Which reduced the training time by a factor of 6.

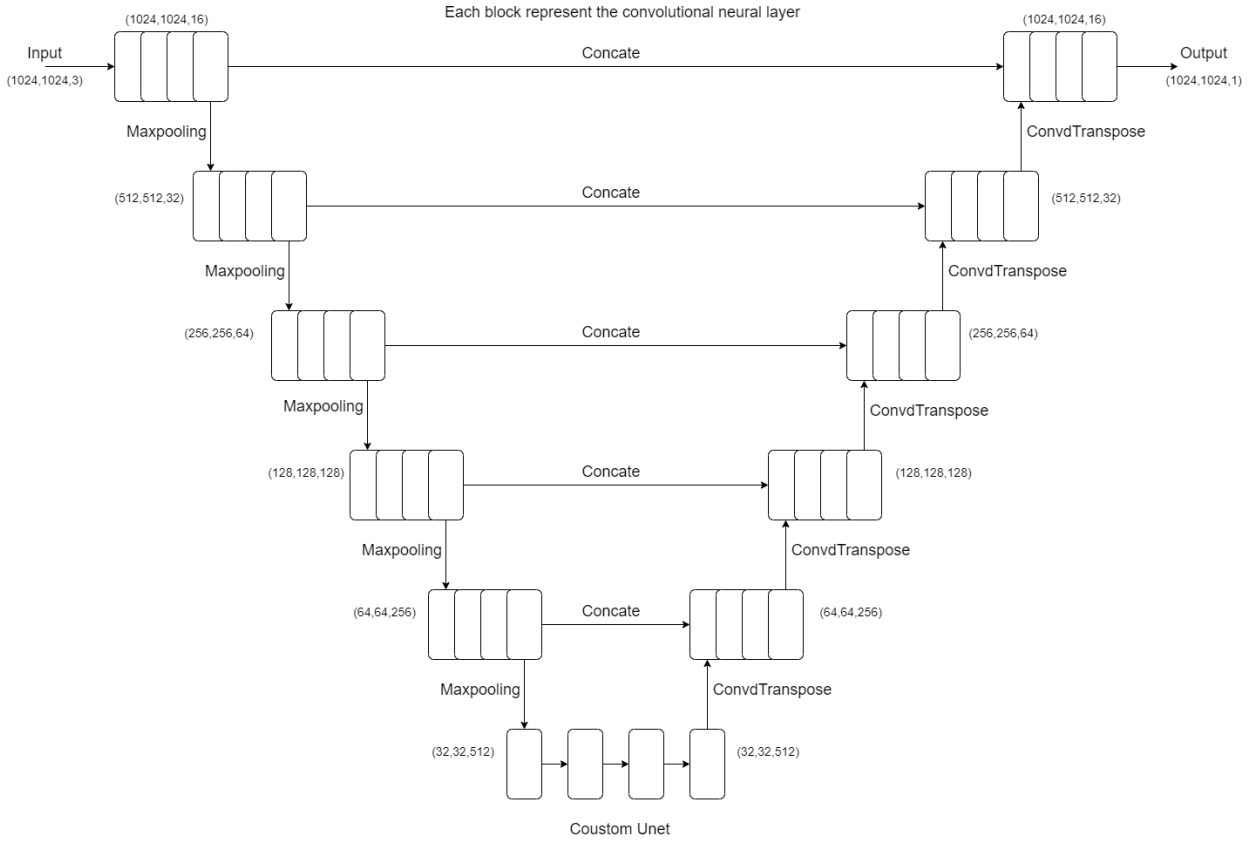


FIG-1 CUSTOM UNET

CUSTOM MODEL 1: The architecture is pretty similar to U-net model but the depth of this model is 2 with single repetition. The depth is 2 which means double upscaling followed by double maxpooling. The advantage of this architecture is that the dimension of input image is (1024,1024,3) which is 4 times the size of image we where passing in custom U-net model (256,256,3). The sequential order of filtters at each depth are (16,32,64). Total Trainable parameters: 349,233

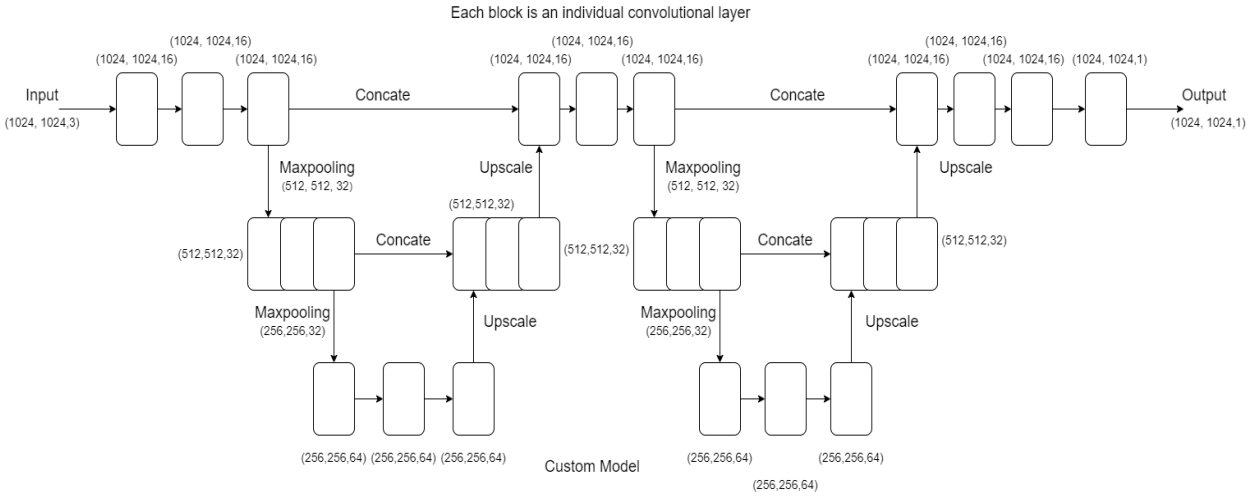


FIG-2 CUSTOM MODEL 1 (DOUBLE U-NET/W-NET)

Custom Model2: The architecture is pretty similar to Custom model 2but the depth of this model is 3 with single repetition and the previous layer's filters of 1 Unet part are passed to the layers of Second U-net at same depth. Similarly, for this architecture also the dimension of input images are (1024,1024,3) which is 4 times the size of image we where passing in custom U-net model (256,256,3). The sequential order of filtters at each depth are (16,32,64,128). Total Trainable parameters: 1,413,585

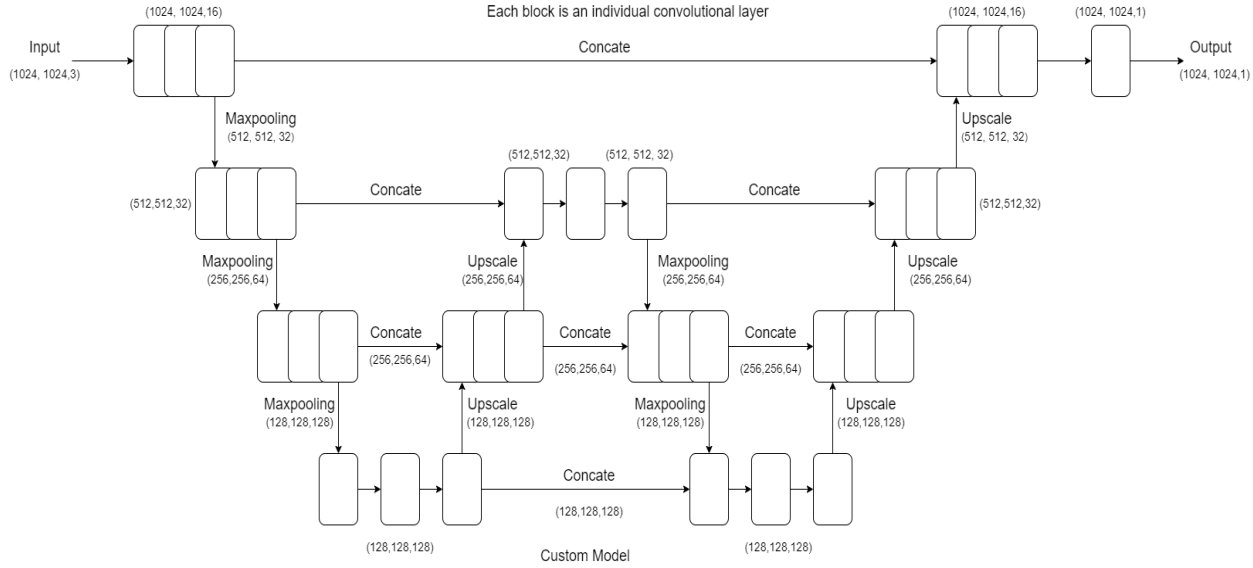


FIG-3 Custom Model 2 (Interconnected W-net)

PROJECT MECHANISM

We ran all the models for 50 epochs to compare them to one another. The dataset was observed for any outliers which might bring down the accuracy. The dimensionality of the images in the dataset was reduced by a factor of 4.5 to reduce the sparsity of the model. All the images were then converted to grayscale from rgb to make the training faster. There was a tremendous amount of time spent on training the models which will compare each model based on the loss function. The dataset was first passed in the Unet build by keras to know how the model was behaving with the Image dataset. After observing the output and the layers of the model and the computation time, we moved on designing our own model which will help us to reduce the tremendous amount of time and space consumed by the Neural Network model.

We designed our Custom Unet model from scratch with 9 blocks which had 3x3 convolution layers and 2x2 max pooling layers and followed by batch normalization we observed the computation time was reduced by a factor of 4. When we were designing the double U-net model, our thought process was, if we are passing the pre-trained model resnet to the U-net then we are getting 89% accuracy. What if we pass the pretrained model again to the U-net model? And also if we reduce the computing complexity of the existing model with the minimal loss in accuracy by reducing the number of filters and depth of the model. As, a result we created a custom model with repeating Unet features(W-net). Similarly, when we are passing the result to the second part of the custom model the features that we extracted at 1st and 2nd depth are not been properly utilize. So, we passed the features to the same depth layers to the second part and created the custom model 2 (internconnected W-net). Which were showing somewhat better result compare to the custom model 1.

We compared the model with the loss function and observed that the Unet with pretrained Resnet works best which is a model created by keras and has almost similar architecture but different specifications which reduced the trainable parameter and reduced the computation time by a factor of 6. We also implemented double Unet which is also a custom build model created from scratch whose architecture is is very simple with 2 depth and single repetition. As, we required to pass input image with size (256,256,3) in existing Unet models or else it was throwing memory error. But for the custom model that we have created was taking input image of size (1024,1024,3) without loosing any features. Which was loss during when we reduce the image size 1500*1500 to 256*256 for Unet. Also, the parameters are around 1,400,000 which are very less compared to the existing model about 55.000,000 parameters. Also the number of filters and depth. These are the factors which reduce teh time and space complexity by much higher rate. For the custom model 2 the input image size i 1024*1024 then applied three convolution layer after every maxpooling and upscaling layers and concatnation of the two layers at same depth after upscaling of image and maxpooling for the second part. The output of the model is greyscale image. So we are converting the Y_train to greyscale for implementing the loss function and metrics.

TEAM MEMBER WORK

[Week 1]: Understanding the Dataset: **Smith and Dhruv**

[Week 2]: Preprocessing the Data: **Smith**

[Week 3]: Extraction of features: **Dhruv**

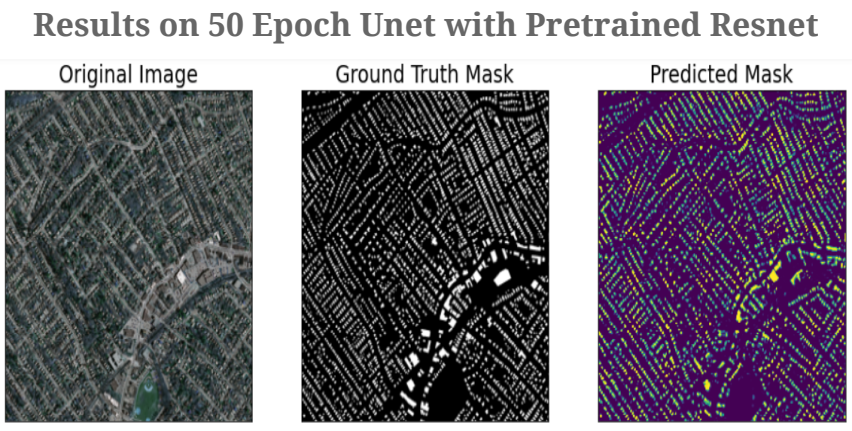
[Week 4]: Creating a machine learning model: **Dhruv**

[Week 5]: Training and fine tuning the model: **Smith**

[Week 6]: Testing the model on testing dataset and working on final report: **Smith and Dhruv**

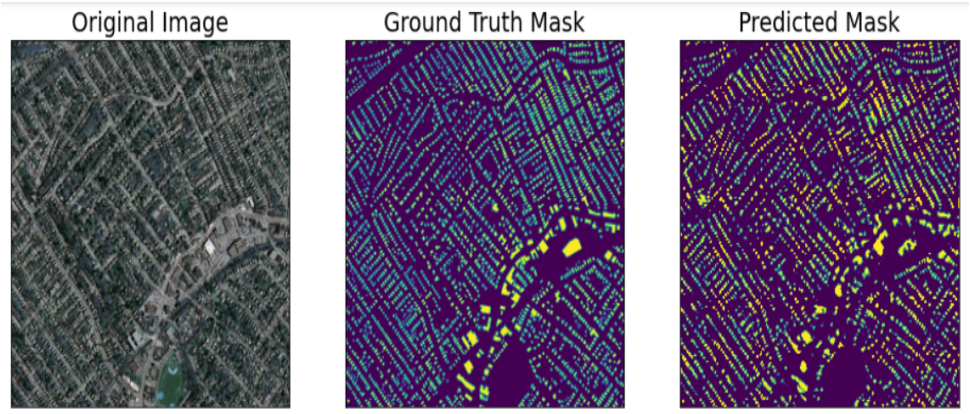
[Week 7]: Accuracy Improvement and final paper writeup: **Smith and Dhruv**

RESULTS



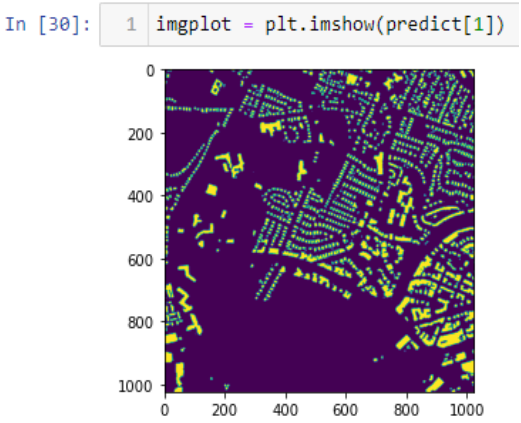
Epoch: 49 dice_loss - 0.1008 Test_accuracy - 89.992
Fig 3 Results using pretrained resnet

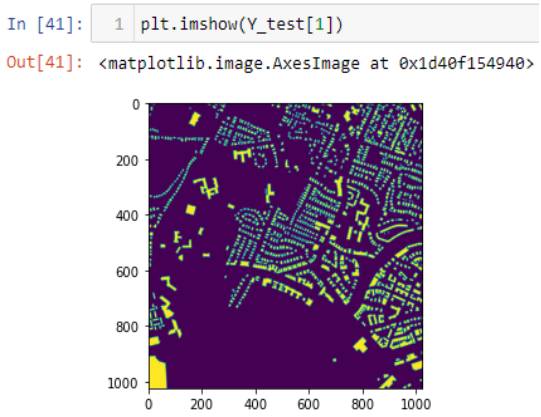
Results with custom Unet with 50 epochs



Epoch: 49 dice_loss - 0.2841 Test_accuracy - 71.59
Fig 3 Results using Custom Unet

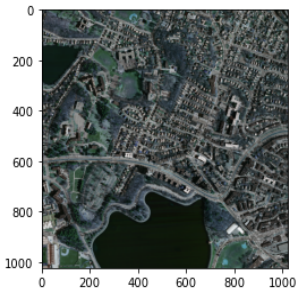
Result with Custom Model 1





```
In [29]:
1 # print(predict)
2 plt.imshow(X_test[1])

Out[29]: <matplotlib.image.AxesImage at 0x1d40b1746a0>
```



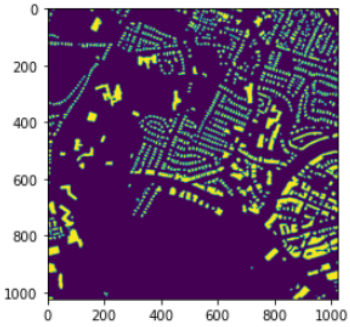
Epoch 49/50: dice_loss = 0.2758, Test_accuracy = 75.19

Fig 4: Results using Custom Model 1

Result with Custom Model 2

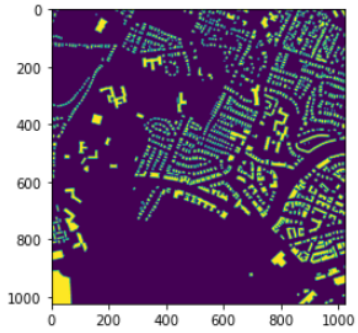
```
In [26]:
1 plt.imshow(predict_4[1])

Out[26]: <matplotlib.image.AxesImage at 0x186881034c0>
```



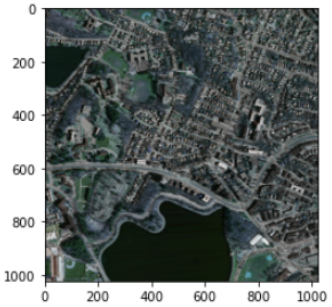

```
In [27]:
1 plt.imshow(Y_test[1])

Out[27]: <matplotlib.image.AxesImage at 0x186881728e0>
```




```
In [25]:
1 # print(predict)
2 plt.imshow(X_test[1])

Out[25]: <matplotlib.image.AxesImage at 0x18687cfb430>
```



Epoch 48/50: dice_loss = 0.2477, Test_accuracy = 75.23

Fig 4: Results using Custom Model 2

QUANTITATIVE ASPECTS OF OUR RESULTS

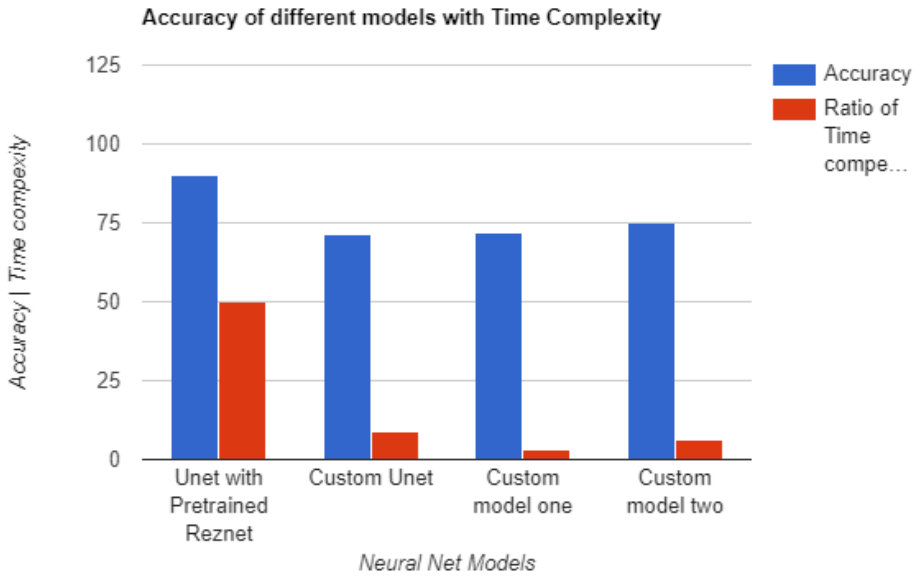


Fig 4 Accuracy of different models with Time Complexity

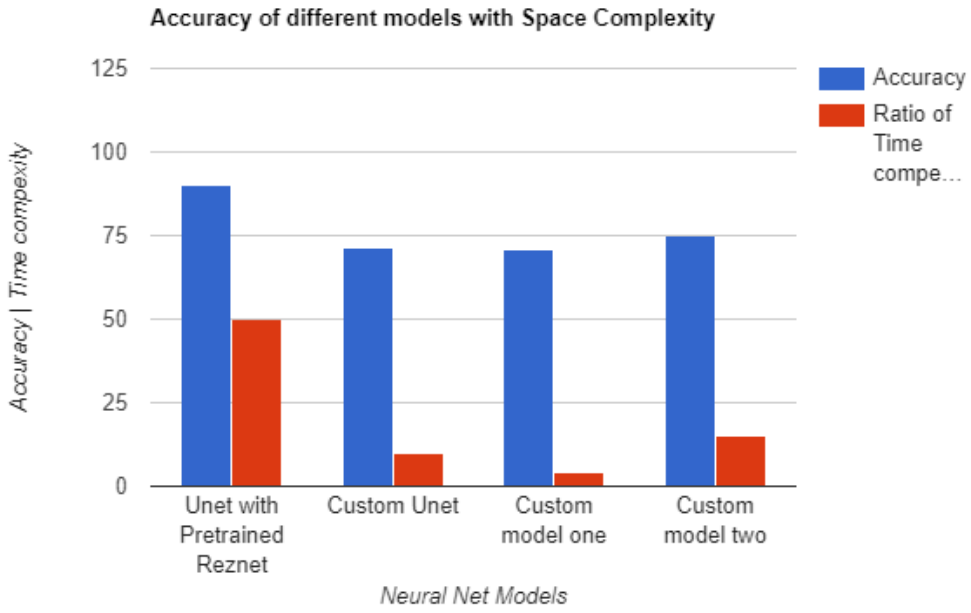


Fig 5 Accuracy of different models with Space Complexity

CONCLUSION

We ran simulations using different neural net models on the massachusett building dataset which has several images in the dataset of size 1500x1500. We saw that the Unet with pretrained resnet was much more time complex and required much space as the parameters were around 50,000,000. When we created our own custom-built Unet model with a reduced filter just enough to get the building boundaries we saw that the time is reduced by a factor of 6 and space by a factor of 5, as the parameter was reduced to around 10,000,000. The W-net was having only 349,233 parameters and accepting images without losing much of its dimension with less number of filters and depth. This implementation was resulted in redusicng time complexity by the factor of 10 and space by factor of 50. Affecting the accuracy by 14% which is quite acceptable. Similarly, for custom model 2 there were 1,413,585 parameters which also reduce the time complexity by the factor of 8 and space by 25 with the salvage of accuracy by 13%.

REFERENCES

1. Volodymyr Mnih, “Machine Learning for Aerial Image Labeling ”, University of Toronto, 2013
2. H. P. Narkhede, “Review of Image Segmentation Techniques”, International Journal of Science and Modern Engineering (IJISME) ISSN: 2319-6386, Volume-1, Issue-8, July 2013.
3. N. Plath, M. Toussaint, and S. Nakajima, “Multi-class image segmentation using conditional random fields and global classification,” in Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009, pp. 817–824.
4. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.
5. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.