[Team 23]Project 2.2: Component Segmentation of Aerial Views/Images

Rajshree Jain
rjain27@ncsu.edu

Sai Shruthi Madhuri Kara
skara2@ncsu.edu

Sruthi Talluri
stallur2@ncsu.edu

1 METHODOLOGY

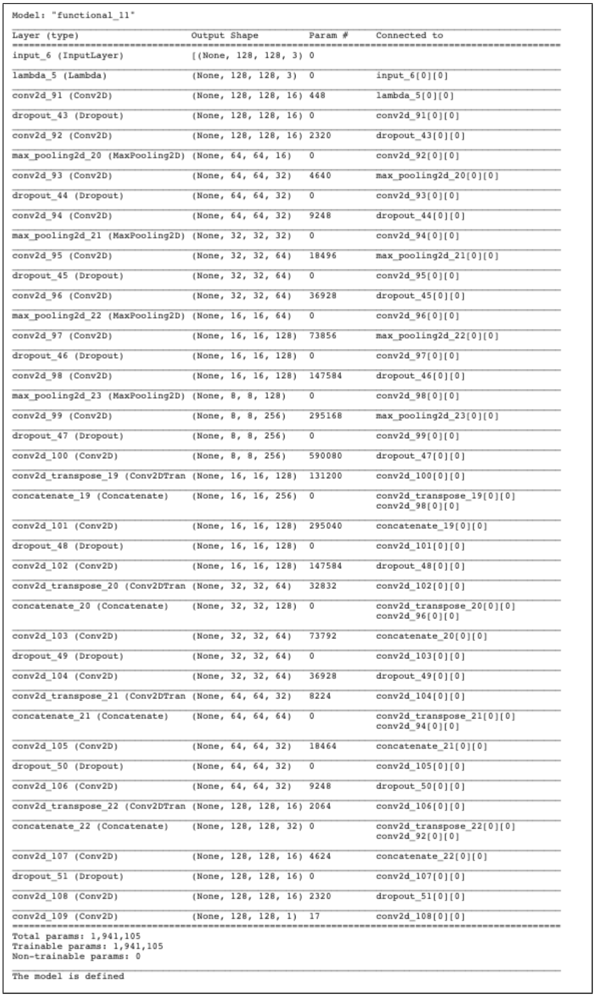
In this project we have worked on a task of Image or Semantic Segmentation which helps us recognize and segregate the different areas in images. We worked on segmenting Ariel images[1] into roads and background. The problem of segmentation can be of two kinds, 1st is image segmentation and 2nd is semantic segmentation[2]. In both the cases each pixel of image is labeled with a particular class which is being represented in the image. Since we try to predict every pixel in the image this task can also be referred as dense prediction. However, the difference between image and semantic segmentation is that in image segmentation we can discriminate different things of same class by giving different labels, however in case of semantic segmentation every item of same class is labeled the same. The task of segmentation of images can be done really well using fully connected convolutional model named UNET.

We have implemented UNET model[3] and its variation to understand which model would perform better in segmentation of images. The UNET model, consists of two paths in which the 1st part is a contraction path(or the encoder part) and the 2nd part is a symmetric expanding path(or decoder path). The 1st path is used to extract the context from the image and the 2nd path is used to precise localizations with transposed convolution.

The most important CNN based operations that are performed in UNET are convolution operation, max pooling operation, and transposed convolution(up sampling of image with learnable parameters). Let us see different layers in the UNET model in detail and how they are useful. In the actual UNET model we first have sets of convolutional layers followed by max pooling layers. Then we have sets of upsamplimg layers followed by convolutional layers. The intution of these layers is that the convolution and pooling layers down sample the image which means the high resolution image is converted to a low resolution image. However, in max pooling we get more information about what is contained in the image. But instead we should get more information of "where" from the image instead of knowing the "what" of the image. For getting the "where" of the image, we up sample accordingly and transposed convolution is a way to up sample and restore to a high sample image.

Please refer to the details of the UNET model in Figure 1, which shows the details of all the layers and also Figure 2, which shows details of a simplified UNET mdoel in which we have removed a set of decoding and encoding layers.

It can be noted[4] that in decoder the size of the image is gradually decreasing and height is increasing as from (128,128,3) to (8,8,256). In case of decoder the opposite happens where the size changes from (8,8,256) to (128,128,1).



```
Model: "functional_11"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_6 (InputLayer)            [(None, 128, 128, 3) 0
__________________________________________________________________________________________________
lambda_5 (Lambda)               (None, 128, 128, 3)  0           input_6[0][0]
__________________________________________________________________________________________________
conv2d_91 (Conv2D)              (None, 128, 128, 16) 448         lambda_5[0][0]
__________________________________________________________________________________________________
dropout_43 (Dropout)            (None, 128, 128, 16) 0           conv2d_91[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, 128, 128, 16) 2320        dropout_43[0][0]
__________________________________________________________________________________________________
max_pooling2d_20 (MaxPooling2D) (None, 64, 64, 16)   0           conv2d_92[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, 64, 64, 32)   4640        max_pooling2d_20[0][0]
__________________________________________________________________________________________________
dropout_44 (Dropout)            (None, 64, 64, 32)   0           conv2d_93[0][0]
__________________________________________________________________________________________________
conv2d_94 (Conv2D)              (None, 64, 64, 32)   9248        dropout_44[0][0]
__________________________________________________________________________________________________
max_pooling2d_21 (MaxPooling2D) (None, 32, 32, 32)   0           conv2d_94[0][0]
__________________________________________________________________________________________________
conv2d_95 (Conv2D)              (None, 32, 32, 64)   18496       max_pooling2d_21[0][0]
__________________________________________________________________________________________________
dropout_45 (Dropout)            (None, 32, 32, 64)   0           conv2d_95[0][0]
__________________________________________________________________________________________________
conv2d_96 (Conv2D)              (None, 32, 32, 64)   36928       dropout_45[0][0]
__________________________________________________________________________________________________
max_pooling2d_22 (MaxPooling2D) (None, 16, 16, 64)   0           conv2d_96[0][0]
__________________________________________________________________________________________________
conv2d_97 (Conv2D)              (None, 16, 16, 128)  73856       max_pooling2d_22[0][0]
__________________________________________________________________________________________________
dropout_46 (Dropout)            (None, 16, 16, 128)  0           conv2d_97[0][0]
__________________________________________________________________________________________________
conv2d_98 (Conv2D)              (None, 16, 16, 128)  147584      dropout_46[0][0]
__________________________________________________________________________________________________
max_pooling2d_23 (MaxPooling2D) (None, 8, 8, 128)    0           conv2d_98[0][0]
__________________________________________________________________________________________________
conv2d_99 (Conv2D)              (None, 8, 8, 256)    295168      max_pooling2d_23[0][0]
__________________________________________________________________________________________________
dropout_47 (Dropout)            (None, 8, 8, 256)    0           conv2d_99[0][0]
__________________________________________________________________________________________________
conv2d_100 (Conv2D)             (None, 8, 8, 256)    590080      dropout_47[0][0]
__________________________________________________________________________________________________
conv2d_transpose_19 (Conv2DTran (None, 16, 16, 128)  131200      conv2d_100[0][0]
__________________________________________________________________________________________________
concatenate_19 (Concatenate)    (None, 16, 16, 256)  0           conv2d_transpose_19[0][0]
                                                                 conv2d_98[0][0]
__________________________________________________________________________________________________
conv2d_101 (Conv2D)             (None, 16, 16, 128)  295040      concatenate_19[0][0]
__________________________________________________________________________________________________
dropout_48 (Dropout)            (None, 16, 16, 128)  0           conv2d_101[0][0]
__________________________________________________________________________________________________
conv2d_102 (Conv2D)             (None, 16, 16, 128)  147584      dropout_48[0][0]
__________________________________________________________________________________________________
conv2d_transpose_20 (Conv2DTran (None, 32, 32, 64)   32832       conv2d_102[0][0]
__________________________________________________________________________________________________
concatenate_20 (Concatenate)    (None, 32, 32, 128)  0           conv2d_transpose_20[0][0]
                                                                 conv2d_96[0][0]
__________________________________________________________________________________________________
conv2d_103 (Conv2D)             (None, 32, 32, 64)   73792       concatenate_20[0][0]
__________________________________________________________________________________________________
dropout_49 (Dropout)            (None, 32, 32, 64)   0           conv2d_103[0][0]
__________________________________________________________________________________________________
conv2d_104 (Conv2D)             (None, 32, 32, 64)   36928       dropout_49[0][0]
__________________________________________________________________________________________________
conv2d_transpose_21 (Conv2DTran (None, 64, 64, 32)   8224        conv2d_104[0][0]
__________________________________________________________________________________________________
concatenate_21 (Concatenate)    (None, 64, 64, 64)   0           conv2d_transpose_21[0][0]
                                                                 conv2d_94[0][0]
__________________________________________________________________________________________________
conv2d_105 (Conv2D)             (None, 64, 64, 32)   18464       concatenate_21[0][0]
__________________________________________________________________________________________________
dropout_50 (Dropout)            (None, 64, 64, 32)   0           conv2d_105[0][0]
__________________________________________________________________________________________________
conv2d_106 (Conv2D)             (None, 64, 64, 32)   9248        dropout_50[0][0]
__________________________________________________________________________________________________
conv2d_transpose_22 (Conv2DTran (None, 128, 128, 16) 2064        conv2d_106[0][0]
__________________________________________________________________________________________________
concatenate_22 (Concatenate)    (None, 128, 128, 32) 0           conv2d_transpose_22[0][0]
                                                                 conv2d_92[0][0]
__________________________________________________________________________________________________
conv2d_107 (Conv2D)             (None, 128, 128, 16) 4624        concatenate_22[0][0]
__________________________________________________________________________________________________
dropout_51 (Dropout)            (None, 128, 128, 16) 0           conv2d_107[0][0]
__________________________________________________________________________________________________
conv2d_108 (Conv2D)             (None, 128, 128, 16) 2320        dropout_51[0][0]
__________________________________________________________________________________________________
conv2d_109 (Conv2D)             (None, 128, 128, 1)  17          conv2d_108[0][0]
==================================================================================================
Total params: 1,941,105
Trainable params: 1,941,105
Non-trainable params: 0
__________________________________________________________________________________________________
The model is defined
```

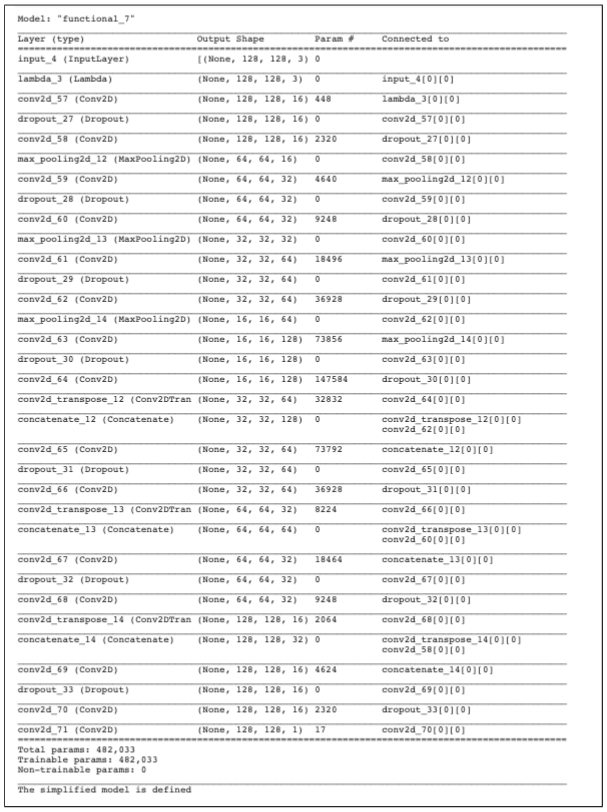
Figure 1: U-Net Model Structure

2 MODEL TRAINING AND SELECTION

2.1 Model Training

(1) Training and Validation Split:
The Massachusetts Roads dataset consists of 1171 aerial images and their corresponding masks. For this dataset the images are randomly split into training set of 1108 images, a validation set of 14 images and a test set of 49 images.

(2) Image and Corresponding Mask:
We have read the input train images, using imread. To read the corresponding mask correctly for the image we have extracted the id from the train image and extracted the mask



```
Model: "functional_7"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            [(None, 128, 128, 3) 0
__________________________________________________________________________________________________
lambda_3 (Lambda)               (None, 128, 128, 3)  0           input_4[0][0]
__________________________________________________________________________________________________
conv2d_57 (Conv2D)              (None, 128, 128, 16) 448         lambda_3[0][0]
__________________________________________________________________________________________________
dropout_27 (Dropout)            (None, 128, 128, 16) 0           conv2d_57[0][0]
__________________________________________________________________________________________________
conv2d_58 (Conv2D)              (None, 128, 128, 16) 2320        dropout_27[0][0]
__________________________________________________________________________________________________
max_pooling2d_12 (MaxPooling2D) (None, 64, 64, 16)   0           conv2d_58[0][0]
__________________________________________________________________________________________________
conv2d_59 (Conv2D)              (None, 64, 64, 32)   4640        max_pooling2d_12[0][0]
__________________________________________________________________________________________________
dropout_28 (Dropout)            (None, 64, 64, 32)   0           conv2d_59[0][0]
__________________________________________________________________________________________________
conv2d_60 (Conv2D)              (None, 64, 64, 32)   9248        dropout_28[0][0]
__________________________________________________________________________________________________
max_pooling2d_13 (MaxPooling2D) (None, 32, 32, 32)   0           conv2d_60[0][0]
__________________________________________________________________________________________________
conv2d_61 (Conv2D)              (None, 32, 32, 64)   18496       max_pooling2d_13[0][0]
__________________________________________________________________________________________________
dropout_29 (Dropout)            (None, 32, 32, 64)   0           conv2d_61[0][0]
__________________________________________________________________________________________________
conv2d_62 (Conv2D)              (None, 32, 32, 64)   36928       dropout_29[0][0]
__________________________________________________________________________________________________
max_pooling2d_14 (MaxPooling2D) (None, 16, 16, 64)   0           conv2d_62[0][0]
__________________________________________________________________________________________________
conv2d_63 (Conv2D)              (None, 16, 16, 128)  73856       max_pooling2d_14[0][0]
__________________________________________________________________________________________________
dropout_30 (Dropout)            (None, 16, 16, 128)  0           conv2d_63[0][0]
__________________________________________________________________________________________________
conv2d_64 (Conv2D)              (None, 16, 16, 128)  147584      dropout_30[0][0]
__________________________________________________________________________________________________
conv2d_transpose_12 (Conv2DTran (None, 32, 32, 64)   32832       conv2d_64[0][0]
__________________________________________________________________________________________________
concatenate_12 (Concatenate)    (None, 32, 32, 128)  0           conv2d_transpose_12[0][0]
                                                                 conv2d_62[0][0]
__________________________________________________________________________________________________
conv2d_65 (Conv2D)              (None, 32, 32, 64)   73792       concatenate_12[0][0]
__________________________________________________________________________________________________
dropout_31 (Dropout)            (None, 32, 32, 64)   0           conv2d_65[0][0]
__________________________________________________________________________________________________
conv2d_66 (Conv2D)              (None, 32, 32, 64)   36928       dropout_31[0][0]
__________________________________________________________________________________________________
conv2d_transpose_13 (Conv2DTran (None, 64, 64, 32)   8224        conv2d_66[0][0]
__________________________________________________________________________________________________
concatenate_13 (Concatenate)    (None, 64, 64, 64)   0           conv2d_transpose_13[0][0]
                                                                 conv2d_60[0][0]
__________________________________________________________________________________________________
conv2d_67 (Conv2D)              (None, 64, 64, 32)   18464       concatenate_13[0][0]
__________________________________________________________________________________________________
dropout_32 (Dropout)            (None, 64, 64, 32)   0           conv2d_67[0][0]
__________________________________________________________________________________________________
conv2d_68 (Conv2D)              (None, 64, 64, 32)   9248        dropout_32[0][0]
__________________________________________________________________________________________________
conv2d_transpose_14 (Conv2DTran (None, 128, 128, 16) 2064        conv2d_68[0][0]
__________________________________________________________________________________________________
concatenate_14 (Concatenate)    (None, 128, 128, 32) 0           conv2d_transpose_14[0][0]
                                                                 conv2d_58[0][0]
__________________________________________________________________________________________________
conv2d_69 (Conv2D)              (None, 128, 128, 16) 4624        concatenate_14[0][0]
__________________________________________________________________________________________________
dropout_33 (Dropout)            (None, 128, 128, 16) 0           conv2d_69[0][0]
__________________________________________________________________________________________________
conv2d_70 (Conv2D)              (None, 128, 128, 16) 2320        dropout_33[0][0]
__________________________________________________________________________________________________
conv2d_71 (Conv2D)              (None, 128, 128, 1)  17          conv2d_70[0][0]
==================================================================================================
Total params: 482,033
Trainable params: 482,033
Non-trainable params: 0
__________________________________________________________________________________________________
The simplified model is defined
```

Figure 2: Simplified U-Net Model Structure

image, hence we were able to read the images correctly in the list. Below are the images(Figure 3) of the train data and the mask for id 1030:

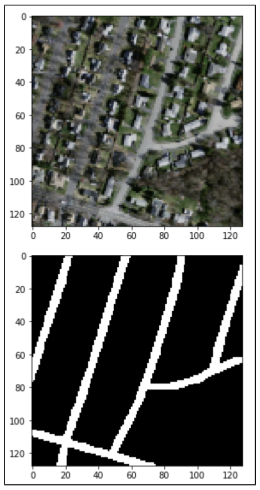


Figure 3: Image, Masked Image showing the Road Segments

Model	Accuracy	Loss	Epochs
U-NET	0.9536	0.1345	32
Simplified U-Net	0.9502	0.1344	25

Table 1: Accuracy, Loss, Epochs for Validation Data

2.2 Model Selection

We considered four major parameters for hyper parameter training when we were training the model. Some are learning rate, optimizer, dropout rate, and number of epochs used during training. First we tried to use stochastic gradient descent and observed the results were not very successful. Further, when we tried using Adam we could observe increased accuracy . The next parameter was dropout rate used within the model. To keep things simple, we have used the same dropout rate in all dropout layers of the model. We tried various rates from no dropout to 0.4 while training the models. We saw better accuracy results when the dropout rate was 0.2. We could observe a increase in error rate when the model used 0.4 dropout instead of 0.2 rate, but this drop was very less. Hence we kept a final dropout rate of 0.2 while training the the model.
We tried different activation functions as the third hyper parameter function. These are the activation functions that convolutional layers would use after every convolution. We tried various activation functions like Elu, Relu etc and found that Relu works better. Then we tried to understand how many epochs would be good to train our model. To understand how many epochs we should run, we tried to train the model for more number of epochs till it over fit. Figure 4 shows an example of a model trained past a point of over fitting. It indicates that the number of epochs that would optimally train a network ranged between 30 and 35.
We have also made the above tuning changes in our Simplfied U-Net model. In the model too we have tried over fitting and see that the number of epochs that would optimally train a network ranged between 20 and 25. In the final images that we get after training and fitting the model, each pixel in the image has a value between 0 and 1. We keep the threshold as .5 in order to decide if that particular pixel belongs to a road or to the background.

3 EVALUATION

The metric that has been used to compare different architectures and understand the correctness of models classification on the test data is validation loss and validation accuracy. The validation loss is best value at 0 and worst value at 1, similarly is the metric accuracy.

In the approach we implemented the U-Net model that was originally used for segmenting biomedical images that contain a larger portion of foreground, or white, pixels, on the roads data compared to the sparse aerial images that the simplified model was segmenting. We compared the loss, accuracy and number of epochs, for determining which one performs better.

We can refer to Table 1 for different metrics calculated for different models. While these results are exciting for the team, it is not confidently believed that the simplified model would outperform the U-Net model as there are only minor differences.

For U-NET model we can see in Figure 7 that the validation accuracy for the validation data is less than the training accuracy. And the validation loss also varies a lot over different epochs initially and then starts increasing,as seen in Figures below. The validation also stabilises and does not improve any further after that. Hence, considering all these factors we have regarded to make an early stopping at 30-35 epochs to stop from over fitting of the model.

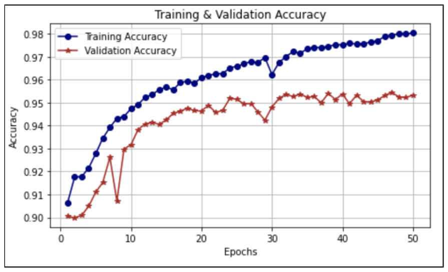


Figure 4: Validation Accuracy for U-NET

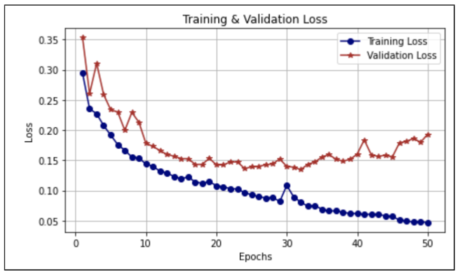


Figure 5: Validation Loss for U-Net

The following are the images for the U-Net model. The generated predictions for the road locations can be found on the bottom row of images found in Figure, while the original images are in the top left of the row and the human annotated masks are in the top right of the row.

In the simplified model of U-Net as mentioned we have removed 4 convolutional layers and created the netweork.

We can see that for this model the validation accuracy is almost equal to that achieved above and the loss is slightly less. But this models achieves best results in lesser epochs of only 20-25, hence helping achieve the results almost similarly at a far better speed. The following plots help us come to this conclusion

The following are the images for the simplified U-Net model. The generated predictions for the road locations can be found on the bottom row of images found in Figure, while the original images are in the top left of the row and the human annotated masks are in the top right of the row.

The models performs best on large roads surrounded by grass or vegetation unlike some of the outlying roads in the corners of the images were surrounded by buildings and parking lots, making it more difficult for the model to correctly predict the location or presence of a road. Even though all these factors, these outputs can be useful in making maps or geographic observations. We believe that a simplified model with lesser (14) Convolutional layers is a better substitute for image segmentation on Ariel data.

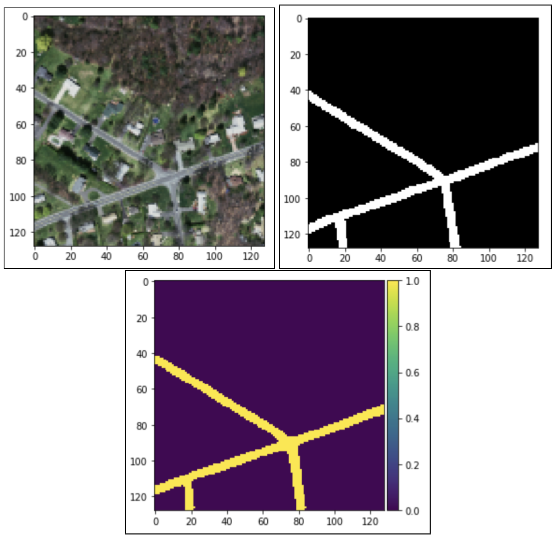


Figure 6: Example of segmented images produced by the model. Top Left: aerial image, Top Right: human annotated mask, Bottom: predicted road locations.

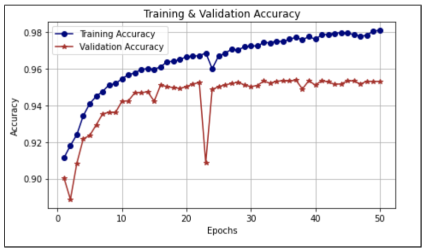


Figure 7: Validation Accuracy for Simplified U-NET

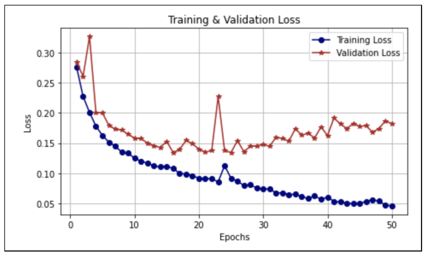


Figure 8: Validation Loss for Simplified U-NET

REFERENCES

[1] V. Mnih, “Machine Learning for Aerial Image Labeling,” PhD Thesis, University of Toronto, 2013.
[2] J.Anumula,\T1\textquoteleftSemanticSegmentationonAerialImagesusingfastai, \T1\textquoteright2019.[Online].Availableathttps://medium.com/swlh/semantic-segmentation-on-aerial-images-using-fastai-a2696e4db127
[3] https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47
[4] https://github.com/hlamba28/UNET-TGS