**Computer Vision Homework 4**

**Abstract**

The scope of the homework is to detect vegetation in some given videos. The method I used is manually labelling some frames and then training a UNet neural network. Finally, I apply the obtained model on every frame in a video and output the resulting video.

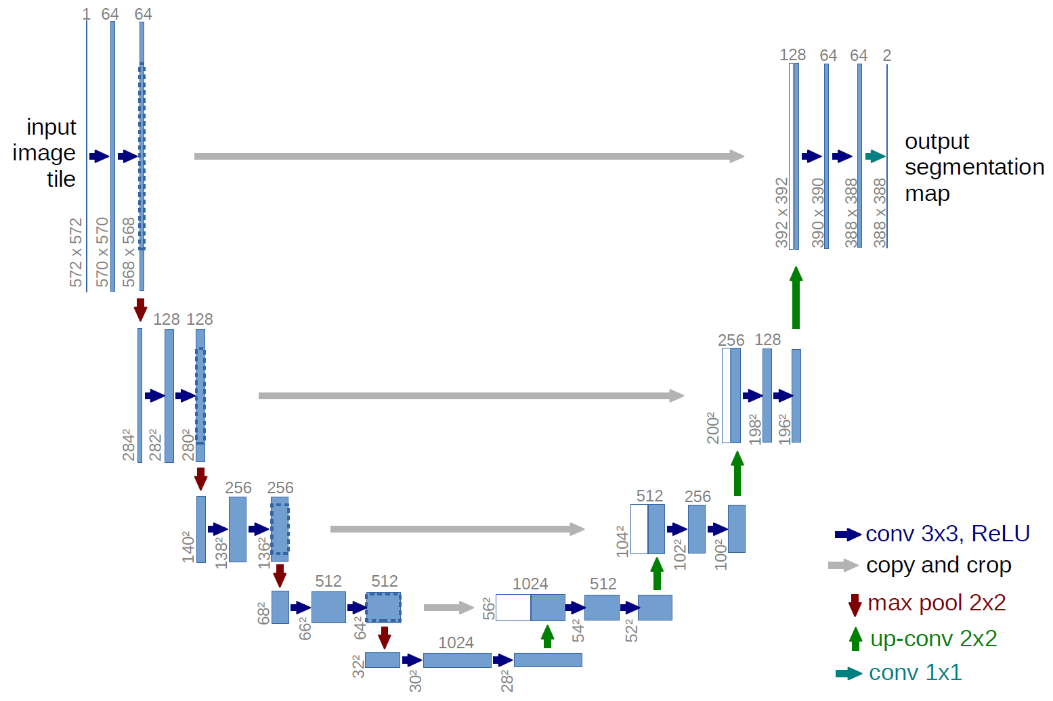
**Files Description**

The videos folders contain the original videos. The dataset folder contains the labelled frames for every video in a separate folder. Each of these folders contain two subfolders named “images”, which contains frames taken every 10 frames, and “masks”, that contains the masks for every respective image. The get\_frames.py file is generating the dataset folder. The get\_masks.py file generates masks from a json file containing points of polygons. The model.py contains the network description. The main.py file trains the models and generates the resulting videos. The results folder contains the resulting videos. The videos with “scene” at the end denote it was obtained by running the scene specific model and the ones with “all” at the end denote it was obtained by running the model built with the data from all the scenes.

**Implementation**

First step was generating frames from the original videos, taken every 10 frames. Then I take these frames and manually label them using the [Make Sense](https://www.makesense.ai/) website. I use the polygon labelling method, which generates a json file with the points of the polygon. Then I had to generate the masks by finding which points in the image are contained in the labelled polygon.

The second step is defining a neural network. I use the UNet architecture since it is built for this type of problem. The architecture looks like this, with the mention that input size is according to our problem:



The last step is training and running the models. I train a model for every specific scene and on that has the data from all the scene. Since we obtain a probability for every pixel to be vegetation, I use a threshold of 0.5 and pixels that have a probability above this number will be labelled as white and those below as black.

I mention that I cut the videos by 8 pixels (from 1920x1080 to 1920 to 1072) in order to easily run the input thought the architecture.

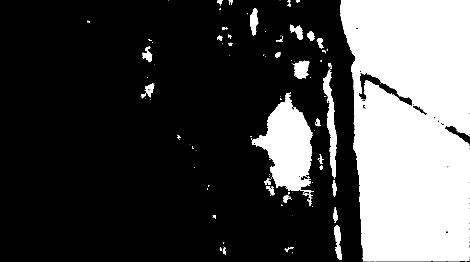
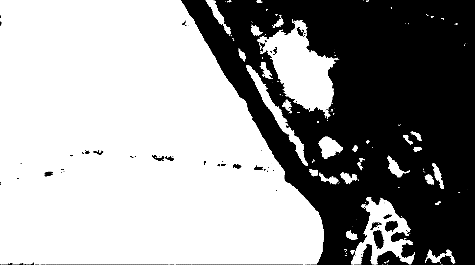
**Results**

Since I tried labelling the vegetation pretty meticulously it took a long time and I didn’t have the time and patience to also label the test frames, so I cannot provide a test accuracy. I will interpret the test results based on the resulting videos.

The results will be displayed a set of three images: the first is the original frame, the second is the result of the scene specific model and the third is the result of the model trained on all scenes.

The training accuracy of the model trained on all the scenes was 0.9553.

1. Train and test on the same scene separately
2. Gura Portitei 1

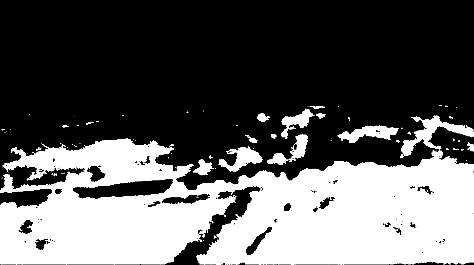
The training accuracy on this scene was 0.9528.

As we can observe, the accuracy is pretty good. As opposed to the classifiers in the previous homework, this model can differentiate between the actual vegetation and the green looking river.

There are no notable differences between the two models, other that the model trained on all the scenes data seems to produce more noise.

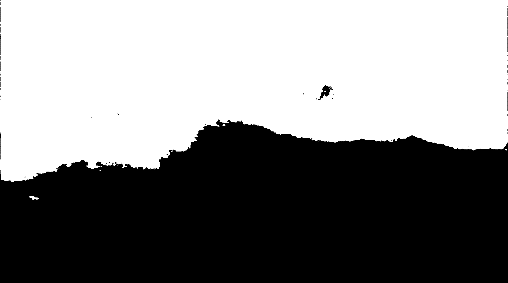
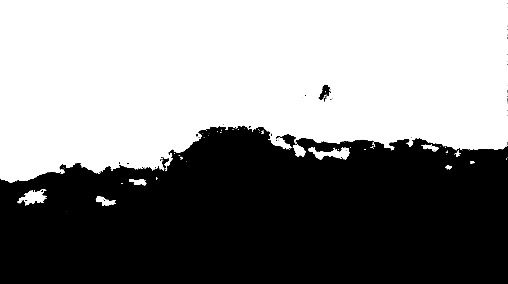
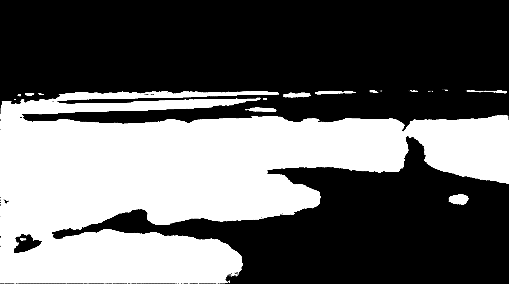
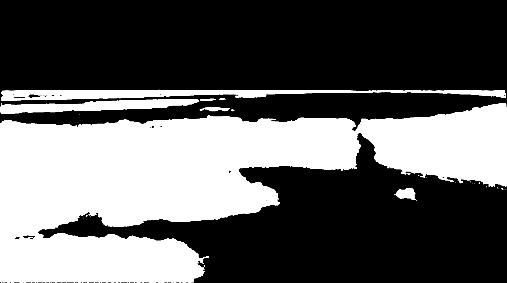
1. Gura Portitei 2

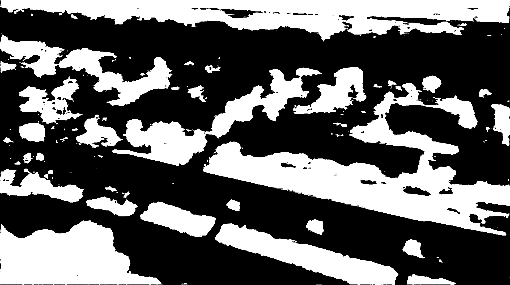
The training accuracy on this scene was 0.9776.



As we can observe the scene specific model overfitted and it produces a simple representation of the vegetation. On the other hand, the model trained on all scenes is able to produce quality detailed segmentation.

1. Delta Crisan 1

The training accuracy on this scene was 0.9570.



We can observe the both models have similar good results. On the last presented scene, the models seem to confuse the green water with vegetation which is rather weird. Perhaps the labelling was off.

**Discussion**

We can observe that the results are much better then those of the previous homework. The model trained on the second scene overfitted, but the model trained on all of the models makes up for it. Given that the labelling was meticulous, the results were satisfying.

**Conclusion**

In conclusion, using the UNet architecture, based on convolutional layer, yielded satisfying results. The labelling was quite laborious work, but in the end, it paid off. Better results can be obtained by having more training examples and an even more meticulous labelling.