Combining deep learning and street view imagery to classify the health of trees

Tom Dicke, 5838533
Nick Dubbeldam, 5647703
Berend Krouwels, 4921909
Wouter Wijkhuizen, 5654971

April 14, 2024

1 Introduction

Soler et al. (Laguarta Soler et al., 2023) propose a new technique using deep learning and street view imagery to map crop types, especially for small farms in developing countries. It utilizes readily available street-level images to train deep learning models for crop identification, eliminating the need for expensive and slow traditional data collection methods. This method successfully created Thailand's first nationwide crop type map, accurately classifying crops like rice and maize. This paves the way for similar large-scale crop mapping efforts globally. Building on Soler et al.'s work with deep learning and street view imagery for crop mapping, this project proposes a similar approach for urban trees in Delft. Instead of identifying crop types, our deep learning models will classify tree health based on street view images. This eliminates the need for expensive, time-consuming traditional data collection methods. To achieve this we converted and reproduced parts of the original paper and code, an overview of the pipeline and our reproductions can be seen in Figure 1. All code can be found on our GitHub https://github.com/nickdubbel/StreetView-CropType.

2 Method

2.1 Dataset

To get the images of the trees, the GPS locations of the tree dataset were sent to the overpass API. With the request to give the way coordinates that are within 25 meters of the trees. This list was filtered to get the points corresponding to roads. From this list, the point closest to the tree was used to retrieve the Street View image. To determine the required heading (orientation) needed to get the tree in frame, the function provided by the original paper was used. This was done for every tree in the data set and all the new coordinates and headings were combined in one .csv file.

This .csv file was used in the provided python script to get the required Street View images. The trees were separated into different dictionaries based on their condition. Examples of the images and their classification can be found in Appendix A. During this process several tree images were lost, 4500+ were lost because they did not have a road point close enough to be used in the dataset. A large part of the remaining trees were lost because the closest road did not have Google Street View data. These were mostly back alleys and small pedestrian roads.

The next step is to process the Street View images, we want to make sure that the model trains on the tree only and not on the background or the colour of the sky for instance. To make this work a new model has been trained. This model is a version of YOLOv8 (Jocher et al., 2023). Where it is retrained on a labelled tree dataset (Detection, 2023). The resulting image is entirely black, except for the tree which remains visible, as in the left image in figure 2.

The model still didn't perform well after implementation. This happens because the model was influenced by the black parts. so we wanted to process the images, even more, to make sure that the black is not influencing the decisions of the model. To do this we segmented the trees from the background, resized them to a square shape, and then resized each tree to a fixed size (320x320 pixels) as seen in Figure 2.

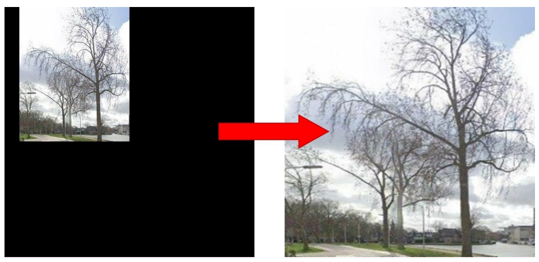


Figure 2: Removing the background of tree images

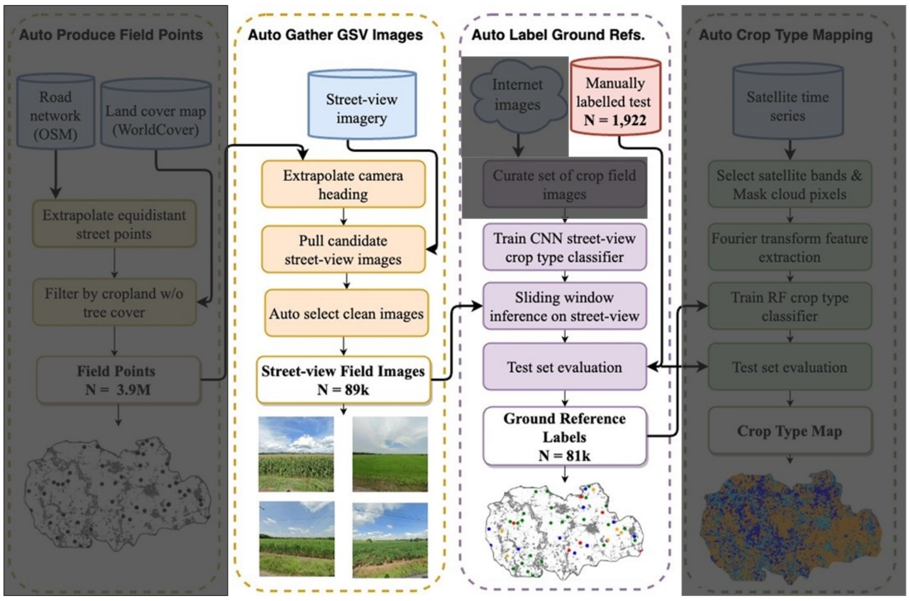

Auto Produce Field Points
Road network (OSM)
Land cover map (WorldCover)
Extrapolate equidistant street points
Filter by cropland w/o tree cover
Field Points N = 3.9M
Auto Gather GSV Images
Street-view imagery
Extrapolate camera heading
Pull candidate street-view images
Auto select clean images
Street-view Field Images N = 89k
Auto Label Ground Refs.
Internet images
Manually labelled test N = 1,922
Curate set of crop field images
Train CNN street-view crop type classifier
Sliding window inference on street-view
Test set evaluation
Ground Reference Labels N = 81k
Auto Crop Type Mapping
Satellite time series
Select satellite bands & Mask cloud pixels
Fourier transform feature extraction
Train RF crop type classifier
Test set evaluation
Crop Type Map

Figure 1: The pipeline of the original paper, the parts that are not reproduced are greyed out

2.2 Model

For the training of the model, the notebook "GSV_CropTypeModel-Train" (thampatron, 2024) was altered in order to train a similar model. For this, the class labels were changed accordingly (tree qualities instead of crop types). We used an existing function, which was commented out in the original code, to calculate the sampling factor. This sampling factor made sure that there is no class imbalance, which could lead to a bias towards the majority class (the class with the largest number of images). Additionally, the image path was changed to point to the dataset with the trees. Lastly, the number of classes were changed, because the tree dataset had more classes. These changes made it possible to train a new model, based on the tree dataset.

3 Hyperparameter tuning

To make the model more efficient, we first processed the pictures into pictures where only the trees could be seen and the rest of the image was padded with zeros. After that, we processed the pictures to go from the images with black to images with only the trees shows and all the pictures were resized to the same size (320x320 pixels). In this section you will find the results of both models and the tuning of the learning rate.

3.1 Blacked out images

The model trained on the full image (640x640) with the background blacked out was trained for 20 epochs which took 20 hours. After this training time we found that it had a poor performance ending up with an accuracy of 0.068 and a global F1 score of 0.067, the accuracy per epoch can be found in Figure 3 and the F1 score per class can be found in Table 2. The large accuracy peaks during training when the model would only choose one of the classes with a large representation, like "matige". The confusion matrix in Figure 4 gives a better insight into the low accuracy and F1 scores, the model tends to give most of the images label 0. This discovery led to our believe that the model did not train correctly on the trees, but rather on the black in the images. Therefore we chose to remove the black from the images as explained in the previous section.

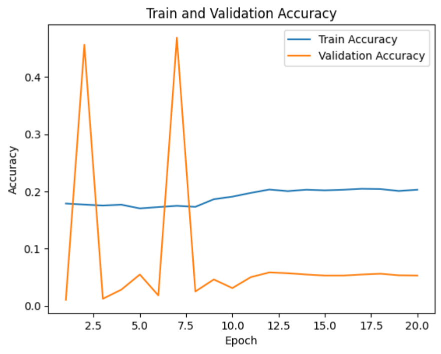


Figure 3: Accuracy of blacked out model

Learning rate	Overal acc	Goed F1	Matig F1	Redelijk F1	Slecht F1	Zeer slecht F1	Dood F1
2E-3	0.2236	0.1827	0.3935	0.	0.	0.	0.0533
7E-3	0.2669	0.	0.	0.	0.4620	0.0339	0.
9E-3	0.2357	0.0400	0.2606	0.1070	0.3600	0.	0.1000
1E-4	0.2960	0.0560	0.2749	0.2682	0.4237	0.	0.
2E-4	0.282	0.0612	0.2129	0.2548	0.4032	0.	0.
2E-5	0.3260	0.1800	0.2615	0.2887	0.4473	0.	0.

Table 1: F1 scores with different learning rates

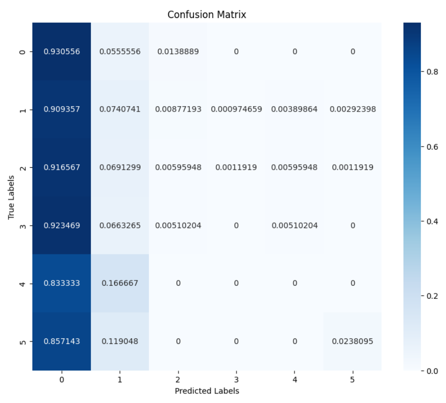


Figure 4: Confusion matrix test data blacked out model

Goed	matige	redelijk	zeer slecht	Dood
0.0646	0.1282	0	0	0.0425

Table 2: F1 score test data blacked out model

3.2 Processed images: Hyperparameter tuning

Now that the data was preprocessed to our liking, as we attained higher accuracies, we could start tuning the learning rate of the model. To find the best learning rate we chose a few starting points and looked at the results to come up with reasonable values to try next, making sure that we covered values in multiple different orders of magnitude. The results are combined into Table 1. The accuracy graphs described over the epochs and the confusion matrices for all the different learning rates can be found in Appendix B.

After the test we can conclude that the model performed best with a learning rate of 2E-5 with the highest overall accuracy of 0.3260 and high F1 scores compared to the other learning rates.

4 Results

After tuning the learning rate we found that the model with learning rate 2E-5 gave the best accuracy and F1 scores. This model was trained on the resized tree images that showed only the tree as described in Section 2.1. The accuracy and F1 scores of this model are shown in the bottom row of Table 1.

5 Discussion

This study aimed to classify tree health in Delft using street view imagery, building upon the methodologies proposed by (Laguarta Soler et al., 2023) for crop classification. While the approach has shown potential, there were several challenges impacting the accuracy and reliability of the model which will be elaborated on below.

Our model had to be trained on a dataset significantly smaller than the dataset used by (Laguarta Soler et al., 2023), this limited the learning capacity and the generalization of our model. Moreover, there were trees in our dataset that were misclassified, this introduced additional noise to our, already quite small, dataset. Also, some classes were underrepresented in our dataset like 'dood' and 'goed', this resulted in a skewed dataset. It might be useful to look into data augmentation techniques, like image rotations and flips, to artificially increase the size of our training dataset.

One challenging aspect of our study was the differentiation between various health states of trees, which proved to be more difficult than distinguishing between crop types like rice fields and corn fields. This makes sense as even for a human identifying tree health requires identifying subtle details while distinguishing between different crop kinds in a field is easier for the untrained eye. To improve the model one might need more advanced feature extraction techniques to easier distinguish between the health of trees.

Finally, the model tends to overfit to the limited amount of training data. To further refine our model, we recommend implementation of regularization techniques to reduce overfitting.

References

Detection, T. (2023, August). Tree detection dataset [visited on 2024-04-12]. https : / / universe . roboflow . com / tree - detection - h9dcy / tree - detection-ekaot

Jocher, G., Chaurasia, A., & Qiu, J. (2023, January). *Ultralytics YOLO* (Version 8.0.0). https://github.com/ultralytics/ultralytics

Laguarta Soler, J., Friedel, T., & Wang, S. (2023). Combining deep learning and street view imagery to map smallholder crop types. *AAAI Publications*.

thampatron. (2024). Streetview-croptype. https : / / github.com/thampatron/StreetView-CropType

Appendices

A Appendix A

This appendix provides examples of each class in the tree dataset to give you a better understanding of the data. The six classes are "Dood" (Dead), "Zeer slecht" (Very bad), "Slecht" (Bad), "Matig" (Moderate), "Redelijk" (Fair), and "Goed" (Good). These images illustrate the visual characteristics that define each class.

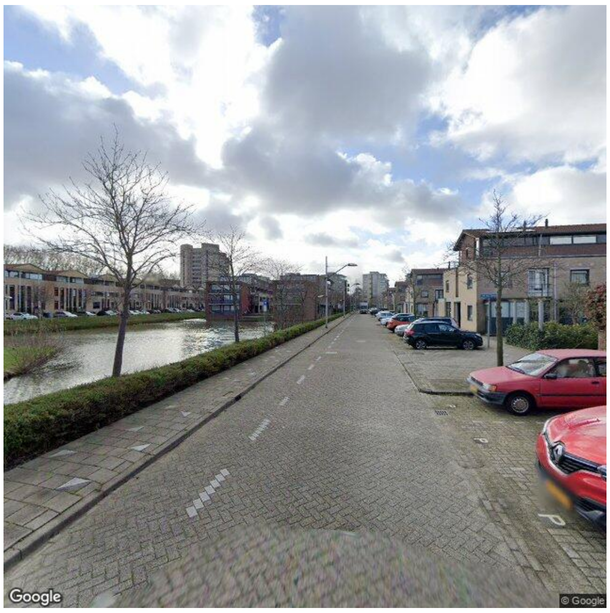


Figure 5: Tree class: "Dood"

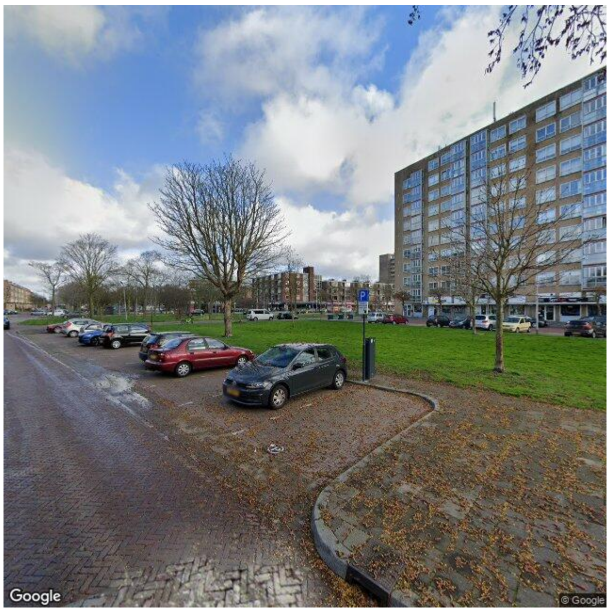


Figure 6: Tree class: "Zeer slecht"

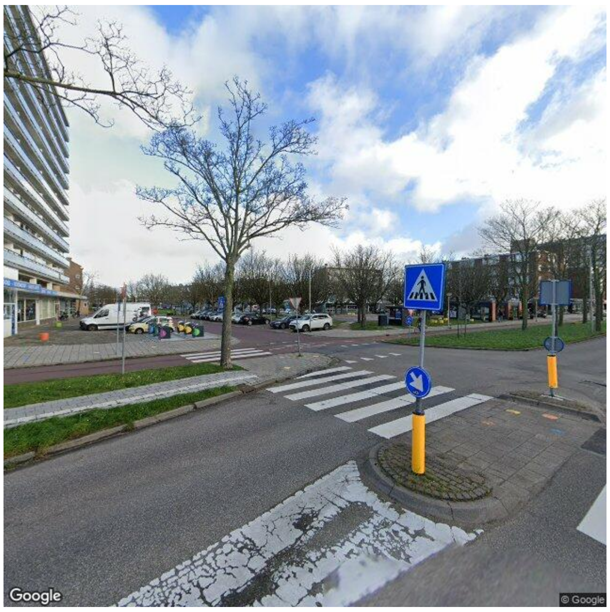


Figure 7: Tree class: "Slecht"

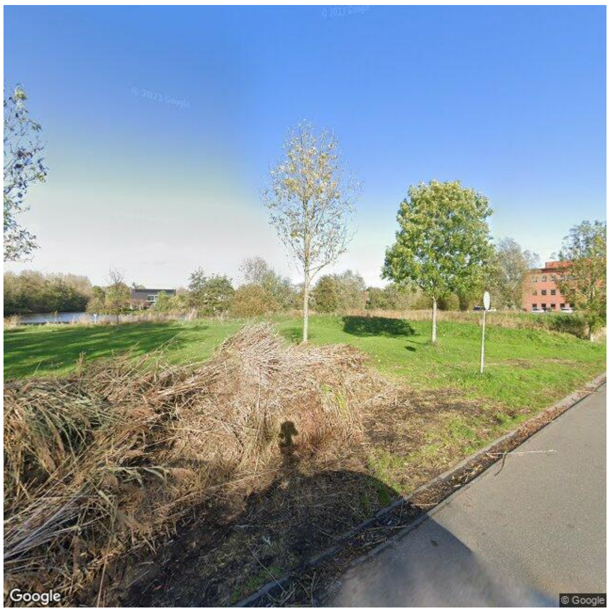


Figure 8: Tree class: "Matig"

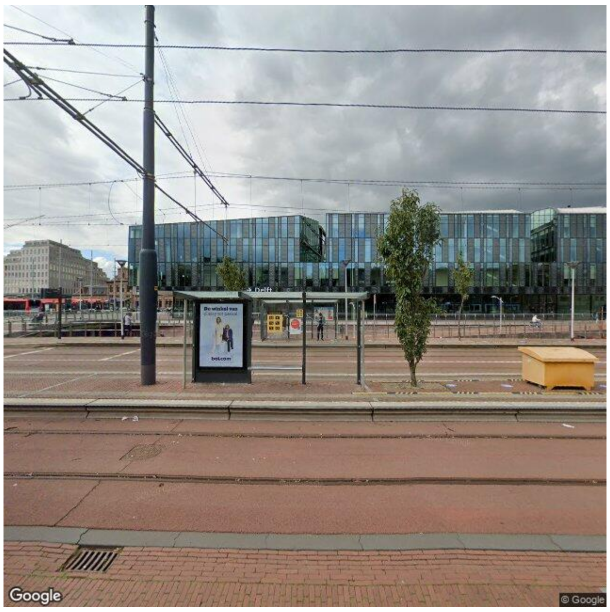


Figure 9: Tree class: "Redelijk"

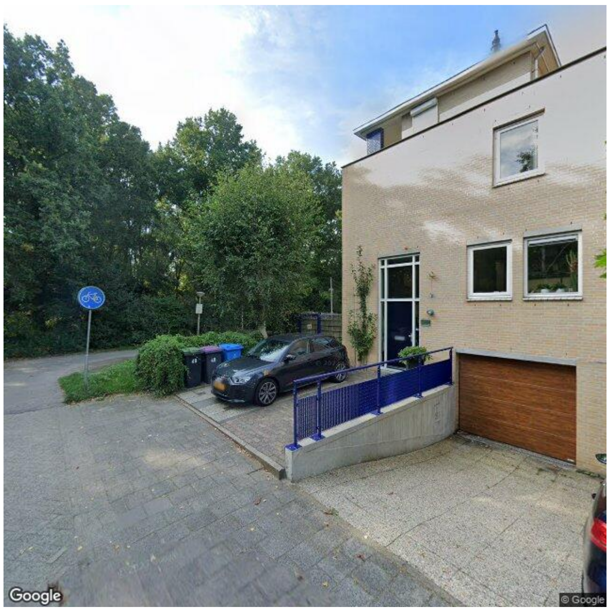


Figure 10: Tree class: "Goed"

B Appendix B

This appendix provides the accuracy graphs described over the epochs and the confusion matrices for all tested learning rates.

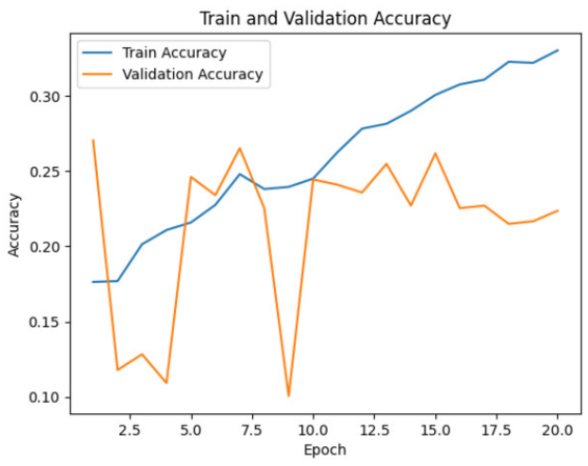


Figure 11: Accuracy of the model with learning rate 2E-3

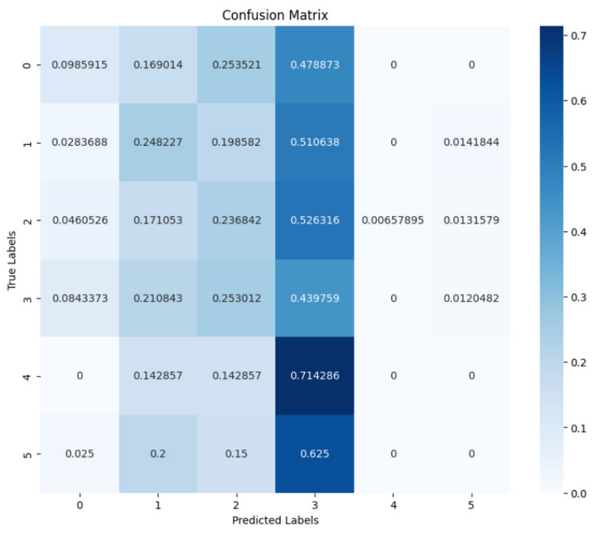


Figure 12: Confusion matrix on the test data of the model with learning rate 2E-3

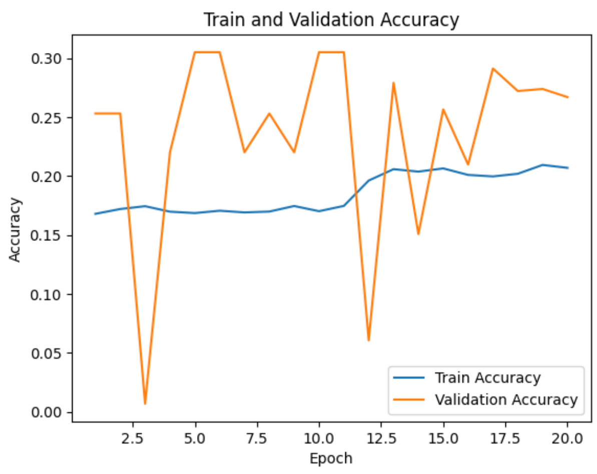
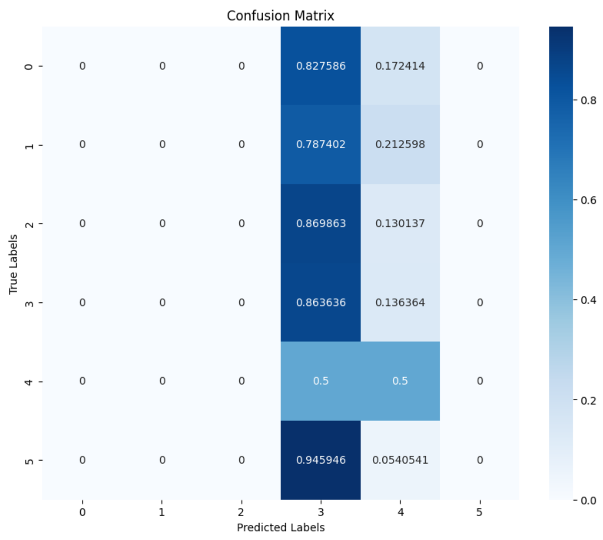


Figure 13: Accuracy of the model with learning rate 7E-3



	0	1	2	3	4	5
0	0	0	0	0.827586	0.172414	0
1	0	0	0	0.787402	0.212598	0
2	0	0	0	0.869863	0.130137	0
3	0	0	0	0.863636	0.136364	0
4	0	0	0	0.5	0.5	0
5	0	0	0	0.945946	0.0540541	0

Figure 14: Confusion matrix on the test data of the model with learning rate 7E-3

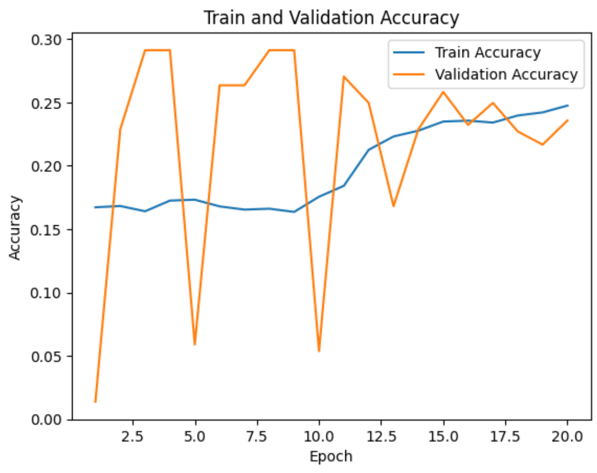


Figure 15: Accuracy of the model with learning rate 9E-3

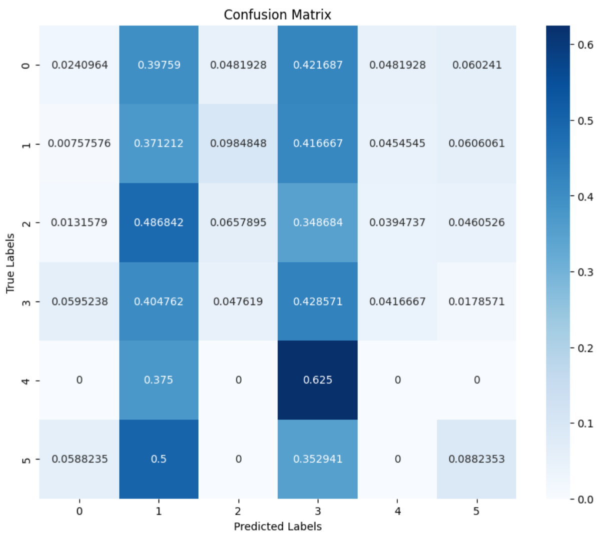


Figure 16: Confusion matrix on the test data of the model with learning rate 9E-3

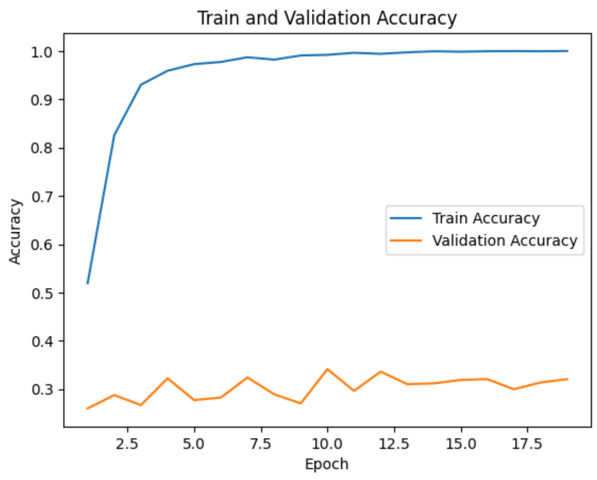


Figure 17: Accuracy of the model with learning rate 1E-4

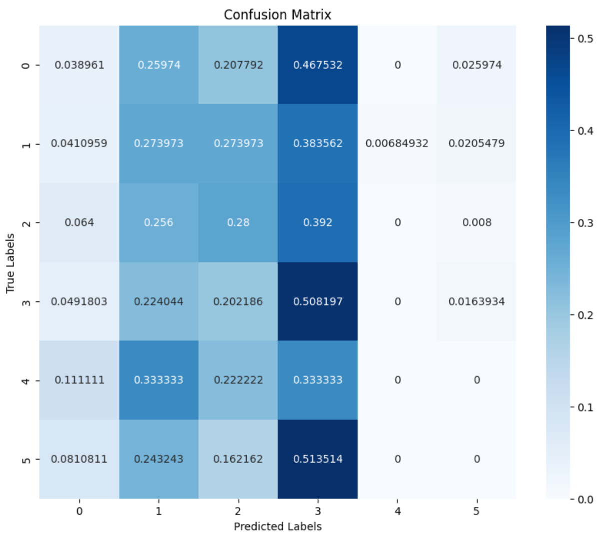


Figure 18: Confusion matrix on the test data of the model with learning rate 1E-4

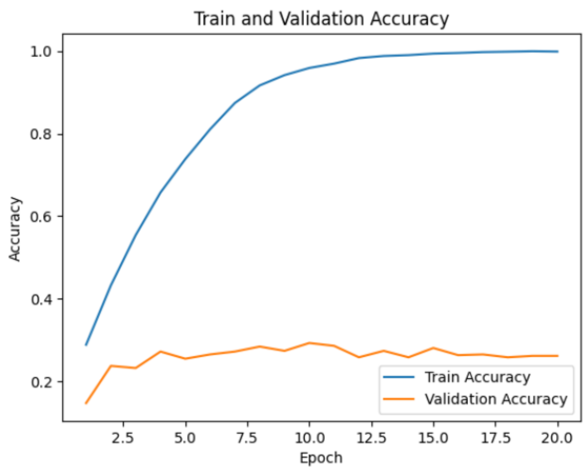


Figure 19: Accuracy of the model with learning rate 2E-4

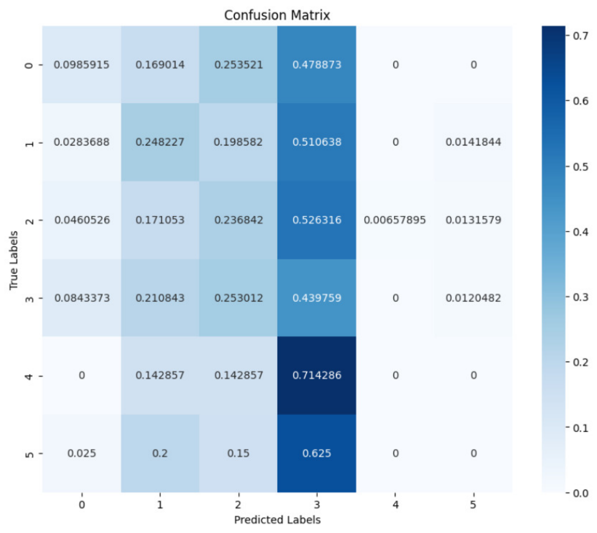


Figure 20: Confusion matrix on the test data of the model with learning rate 2E-4

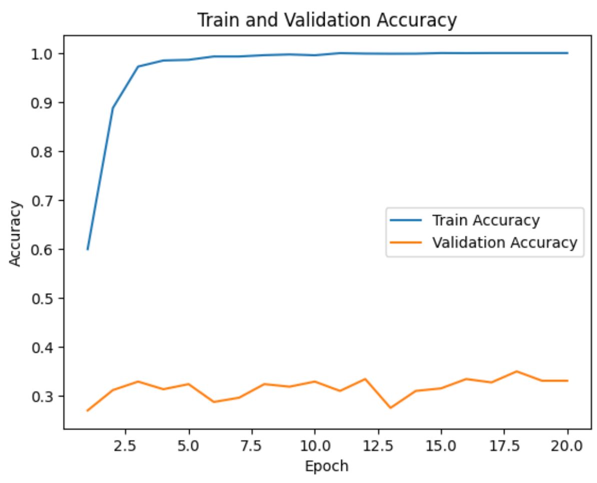


Figure 21: Accuracy of the model with learning rate 2E-5

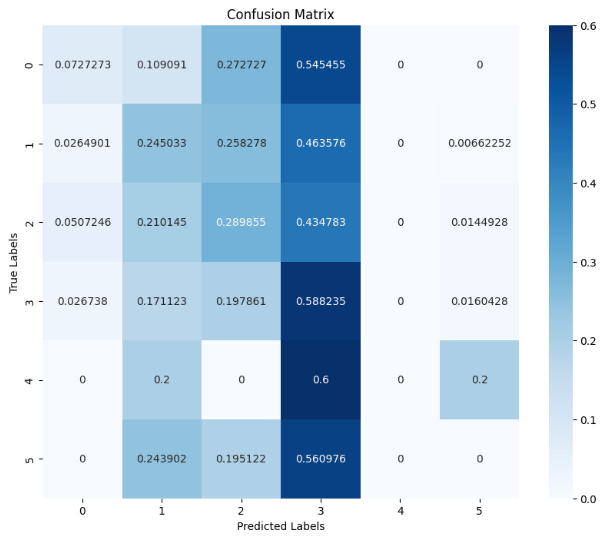


Figure 22: Confusion matrix on the test data of the model with learning rate 2E-5

C Appendix C

In this appendix we describe who did what part of this project.

Tom:

- Co-wrote the project plan, made the excel planning.
- Rewrote the code to train the network (together with Nick)
- Wrote the code to go from images padded with zero's to images with only the trees shown with all an equal size.
- Trained the model for a learning rate of 2E-4 and 2E-3.
- Co-wrote the final report: Introduction, subsection about the model (with Nick), Hyper parameter tuning, Results (with Nick), Discussion.

Berend:

- Co-wrote the project plan
- Explored possibilities for image normalization
- Explored possible tree detection models
- Co-wrote the final report (Method (Dataset), Results, Discussion & finalizing the report)

Wouter:

- Co-wrote the project plan
- Modified original code for getting road points and street view images.
- Trained the model for detecting trees in images.
- Trained the model on blacked out images of trees
- Co-wrote the final report.

Nick:

- Co-wrote the project plan
- Wrote the code to train the tree detector (YoloV8)
- Wrote the code to process all the images and make the background black with the tree detector.
- Rewrote the code to train the network (together with Tom)
- Trained the model for a learning rate of 2E-5 and 1e-04
- Co-wrote the final report: section about detecting the trees (with Berend), subsection about the model (with Tom) and Hyper parameter tuning (with Tom)