Semantic Segmentation for Forest Aerial Views

Machine Learning in Intelligent Transportation: Final project

Adam Cihlar[1]

22. June 2023

Abstract: In the context of transportation, pilots need to identify possible urgent landing areas in forests. In this work we develop a semantic segmentation model based on U-Net architecture. The selected model achieves 0.9084 Intersection over Union metric on held-out test set and can be applied in real-time settings as it processes over 45 FPS even if running only on a CPU.

Keywords: Semantic segmentation, U-Net, SegFormer

Introduction

Semantic segmentation models have experienced a great development and success in the recent years being applied in large variety of different fields ranging from medical diagnostics to applications in autonomous driving. In this work we will develop a model for semantic segmentation of forest aerial images to identify a possible urgent landing areas.

The structure of the paper goes as follows. First, we introduce the data, the preprocessing and augmentation techniques that we apply. Then, we will describe the selected models and training procedures that we utilize to optimize the single models, and depict the whole flow in a diagram. In chapter 3 we report the results of the models and visualize their predictions on unseen data. Finally, we elaborate on the achieved performance.

[1]University of Klagenfurt, Faculty of Technical Sciences, Department of Smart Systems Technologies, Degree: Information and Communications Engineering, adcihlar@edu.aau.at

1 Data and preprocessing

The dataset at hand is a collection of aerial photos of a forest and accompanying image masks that indicate the presence of birch trees in the images. These masks are binary, with a value of 0 representing areas without birch trees and a value of 1 indicating the presence of birch trees. Our objective is to utilize this dataset to train a model capable of performing semantic segmentation, accurately identifying and delineating birch trees in the images.

All images have a size of 128×128 pixels and are in RGB format; thus, having three channels. The size of the binary masks corresponds to the size of the images.

The training set containing both the samples and respective masks consists of 4000 samples. The test dataset contains 1000 samples, but does not include ground truth masks. Therefore, we split the original training set to training, validation and test sets in 60:20:20 ratio. The first two serving for development and optimal choice of hyper-parameters of the individual models, the latter for comparison across the fine-tuned models.

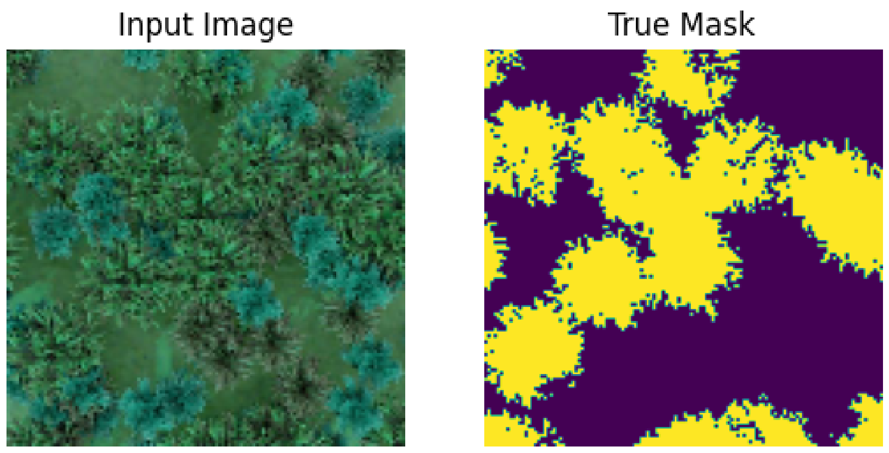


Figure 1: Sample from the training set

To increase the amount of training data, we utilize several data augmentation techniques. As the meaning of the picture does not change after rotation or flipping, we apply random rotation, horizontal and vertical flipping and reflection. To ensure that the model is robust to varying light conditions, we apply also random change in brightness and contrast of the images. Each of the augmentations is applied with probability 0.5. First, for searching the optimal hyper-parameters, we extend only the training set. In the second step after the hyper-parameters search, we augment also the validation set. A sample of an augmented image may be found in appendix (figure 4).

2 Models and training procedures

We utilize two different model architectures and develop in total 4 models. For the first two models, the underlying architecture follows the U-net architecture developed by Ronneberger et al. (2015). In the third model, we transform a pretrained VGG16 model (Simonyan et al., 2014) to the U-Net architecture. To experiment also with different approach, we choose SegFormer (Xie et al., 2021) as the last model.

2.1 U-Net

The U-Net architecture is a convolutional neural network (CNN) model, which consists of an encoder path that captures spatial information and a decoder path that enables precise localization of the identified features.

The basic building unit of the model is convolutional block consisting of two successive 2D convolutions with ReLU activation functions. The model consists of encoder and decoder part and in total of 9 of the convolutional blocks. In the encoder the blocks are followed by max-pooling operation and dropout, in the decoder they are followed by transposed convolution and dropout. The first and last, second and eight, third and seventh and fourth and sixth blocks are connected with copy and crop operation in the decoder part, which significantly enhances the models performance as it helps to restore the original position of the identified features. The encoder and decoder are connected through a bottleneck layer consisting of two convolutional layers with a smaller filter size. The purpose of this layer is to capture the most relevant features for segmentation. The decoder path concludes with a final convolutional layer that produces the segmentation map, since we are dealing with binary classification, we apply sigmoid activation function on top of the final layer.

We experiment with various hyper-parameter settings. Namely, we perform a grid search over the following hyper-parameter space:

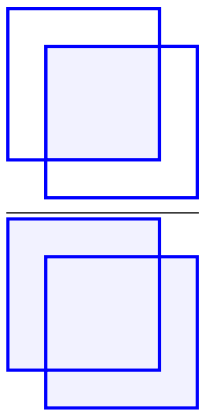
- Number of filters $\in$ [8, 16, 24, 32, 48]
- Dropout $\in$ [0, 0.2, 0.4]
- Learning rate $\in$ [0.005, 0.001, 0.0005]

We optimize the models using Adam optimizer (Kingma et al., 2014) in its original settings, except the learning rate, through minimizing the dice loss function.

$$\text{Dice Loss} = 1 - \frac{1}{c}\sum_{i=0}^{c} \frac{\sum_j^N 2y_i^j \hat{y}_i^j + \varepsilon}{\sum_j^N y_i^j + \sum_j^N \hat{y}_i^j + \varepsilon} \tag{2.1}$$

We train the models for maximum 12 epochs and take the model from the epoch with best result on validation set. The best hyper-parameters are selected based on the Intersection over Union metric (IoU) on validation test set.

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \tag{2.2}$$



However, we also track the accuracy and F1-score of the models. The best results of the individual models on validation test can be found in appendix in table 3. As assumed, the performance of the model rises with number of filters in the convolutional layers. Therefore, for the further retraining on training and validation sets, we select the best performing model with 48 filters and the best performing model with 8 filters to analyze the trade-off between size and efficiency of the models.

Both selected models are first retrained on the training set only but for 50 epochs to find out the optimal number of epochs. The optimal number of epochs is 29 for the larger model reaching $\text{IoU} = 0.9184$, and 8 epochs for the smaller with $\text{IoU} = 0.8853$, respectively. Then, we augment also the validation set and use both data sets (training and validation) for the final training. The results on unseen test set will be evaluated together with the two other models in section 3.

2.2 VGG16 U-Net

For the third model, we utilize a VGG16 model (Simonyan et al., 2014) pretrained on ImageNet dataset as the encoder part of the U-Net. Therefore, the decoder part needs to be adjusted to conform to the encoder, so that we can apply the crucial connections between encoder and decoder. The resulting model has over 30 million parameters, so we freeze the pretrained encoder part and only train the randomly initialized decoder to make the training more feasible. The last layer is again followed by a sigmoid activation function.

The training procedure follows the same scheme as for the previous U-Net models. We optimize the model by minimizing the dice loss function with Adam optimizer and take the learning rate which was optimal for the U-Net model with 48 filters. The model is first trained for 50 epochs on the training set and then retrained using the optimal number of epochs on training and validation sets combined. We reached the best IoU metric (0.9002) on the validation set after 25 epochs.

2.3 SegFormer

Segformer (Xie et al., 2021) builds on the transformer architecture proposed by Vaswani et al. (2017) and utilizes the attention mechanism in the encoder part of the model. The encoder is followed by a simple multi-layer perceptron decoder that aggregates information from different layers of the encoder, and thus combines both local and global features which results in powerful representations. To support the dense prediction task, the input image with original dimensions HxWx3 is divided to small 4x4 patches. This approach promotes dense predictions by capturing more local details. These patches are then fed into the hierarchical Transformer encoder, which produces multi-level features at resolutions of {1/4, 1/8, 1/16, 1/32} relative to the original image size. These multi-level features are subsequently passed to the MLP decoder, which predicts the segmentation mask at a resolution of $\frac{H}{4} \times \frac{W}{4} \times C$, where C represents the number of classes. Therefore, from the nature of the model, the masks cannot capture the finest features, which by default favours the previous models as the masks in our dataset are very detailed, capturing single branches of the birch trees.

We start the training from a small version of SegFormer (3.72 million parameters) pretrained on ADE20k semantic segmentation dataset but adjust the final layer to conform to our binary masks. Then, we fine-tune the whole model on our training data for 30 epochs and choose the optimal number of epochs. For the training hyper-parameters, we follow the original paper and optimize the model with Adam optimizer with learning rate of 0.00006 and batch size 16. Finally, we retrain the model for the selected number of epochs (21) on augmented training and validation data.

The learning curves for all of the models may be found in appendix (figures 5, 6, 7 and 8). The diagram of the whole pipeline, including data preprocessing, augmentation, training and evaluation, is shown in figure 2.

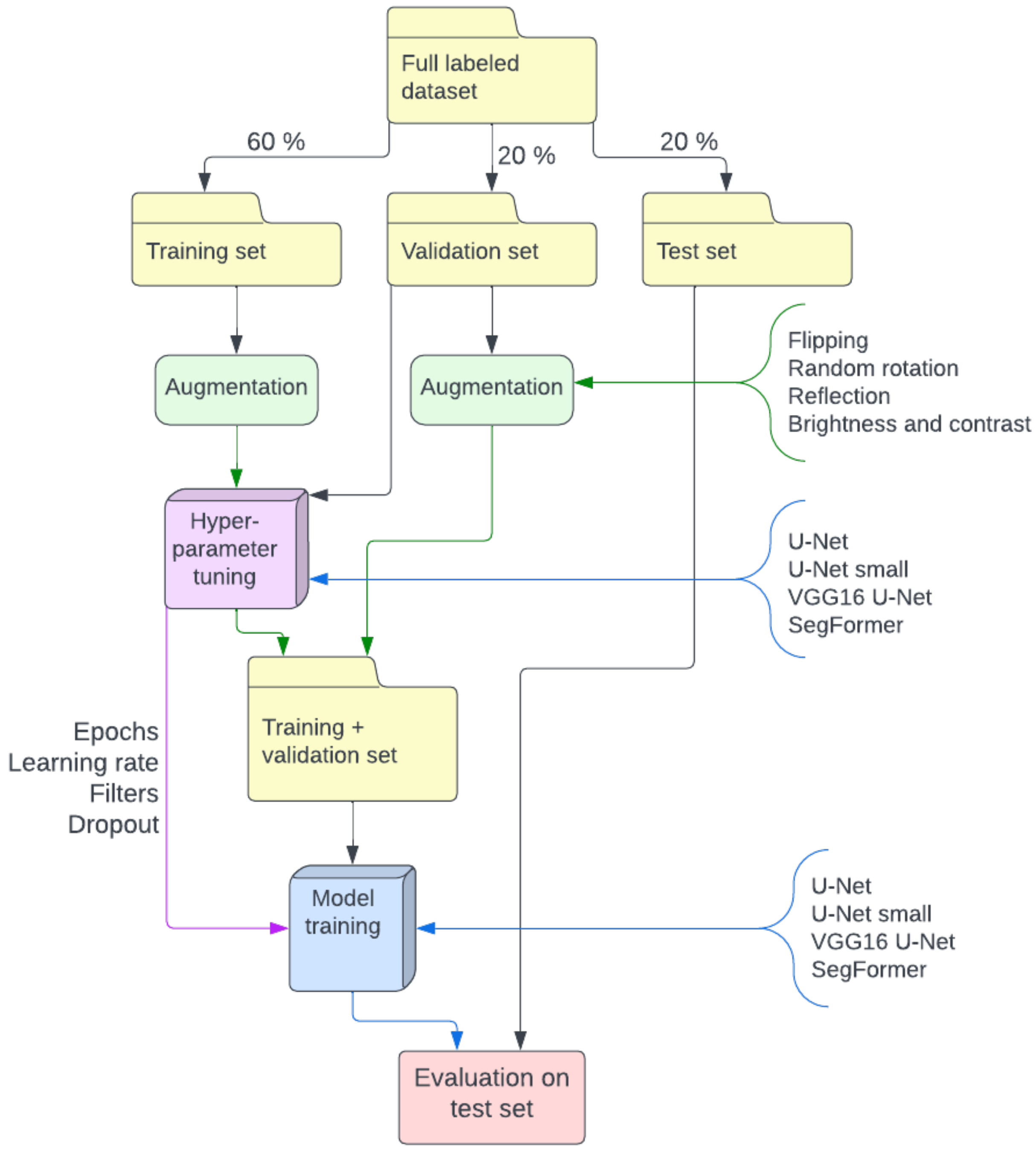


Figure 2: Diagram of the pipeline

Note. Yellow components represent data, green components represent instances of augmentation algorithms, the purple box depicts hyper-parameter tuning, the blue one training of a model and red box stands for evaluation of a trained model. Black arrows depict the flow of raw data, green arrows of augmented data, purple arrow transfers the optimal hyper-parameters and blue arrows stands for flow of the models.

3 Evaluation

In this section, we will present how the four developed models compare to each other on unseen test dataset. We evaluate the performance using accuracy, intersection over union and f1-score. For qualitative comparison, we also display one image from the test set, its ground truth mask and predictions of the individual models on that image.

Model	Parameters	(trainable)	Dice Loss	Accuracy	IoU	F1 score
U-Net	19.43 M	(19.42 M)	0.0334	0.9597	0.9219	0.9558
U-Net small	0.54 M	(0.54 M)	0.0395	0.9523	0.9084	0.9478
VGG U-Net	31.95 M	(17.23 M)	0.0423	0.9494	0.9030	0.9446
SegFormer	3.72 M	(3.71 M)	0.1104	0.9082	0.8010	0.8895

Table 1: Comparison of models on test set

Note. We report the dice loss used for training the model. As we specified the loss as $DiceLoss = 1 - DiceCoefficient$, we omit the dice coefficient in the presented table to avoid redundancy.

According to the reported metrics, the SegFormer model achieves significantly worse results than all other models. We argue, that the model is not suitable for the task at hand, because the granularity of its prediction is not a single pixel but patches of pixels of size 4×4. As the model produces prediction of insufficient quality, we do not consider it for further comparison.

The other three models based on U-Net architecture achieved significantly better results. The larger version of U-Net model with 48 filters, zero dropout and learning rate of 0.0005, trained for 29 epochs is superior to the other models. However, at the same time it has the biggest amount of trainable parameters. Even though the U-Net model consisting of pretrained VGG16 encoder has by far the most parameters in total, as we froze the encoder, it has slightly less trainable ones than the U-Net model. The training times on single GPU of both of the networks are comparable with 34 minutes for the first and 29 minutes for the latter, respectively.

However, the smaller version of U-Net, which consists of only 8 convolution filters per layer and has almost 40 times less trainable parameters than the previous two models, achieved very competitive results with the larger U-Net and even slightly better results compared to the VGG16 U-Net. The training time took around 15 minutes as we were training the model for 88 epochs. However, a substantial difference can be observed in inference times, as the small U-Net processes around 1080 images per second, while the large U-Net around 200 images per second and VGG16 U-Net approximately 130 images per second, on single GPU. Clearly, with this computational resources, all the three models could be used in real time settings. We also compare the inference times on CPU. In this case, the only model satisfying a requirement of processing at least 30 frames per second, which we define as sufficient for real time video processing, is the U-Net small model. The overview of training and inference times may be found in table 2

Model	Parameters	(trainable)	Training time (GPU)	Inference (GPU)	Inference (CPU)
U-Net	19.43 M	(19.42 M)	34 min	200 FPS	3.6 FPS
U-Net small	0.54 M	(0.54 M)	15 min	1080 FPS	45.5 FPS
VGG U-Net	31.95 M	(17.23 M)	29 min	130 FPS	1.8 FPS

Table 2: Comparison of training and inference times

To compare the performance of all the models also visually, we demonstrate the predictions on randomly selected image from the test set (figure 3). Conforming to the presented metrics, we can hardly distinguish the quality of the predictions of the three U-Net based models, while there is a clear lack of details captured by the SegFormer.

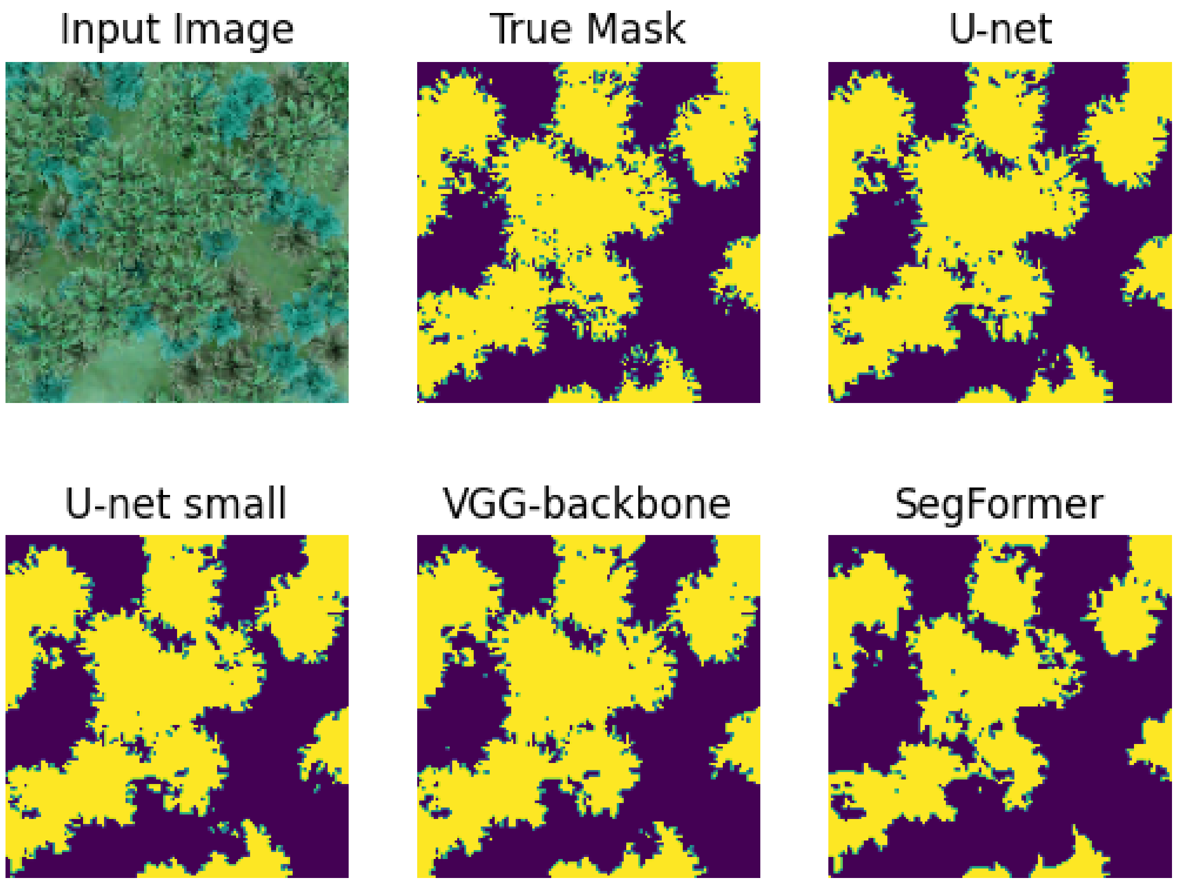


Figure 3: Comparison of the trained models on a random test sample

Conclusion

In this work we trained four models for semantic segmentation of aerial forest views. First, we inspected the data and augmented them using random rotation, horizontal and vertical flipping, reflection and brightness and contrast adjustment, to get more samples for training. In the second chapter we introduced all the considered models and procedures that we utilized to train them. In thee last chapter, we evaluate the models on held-out test set and comment on their performance and computational costliness.

Models based on U-Net architecture achieved very comparable results and all could be used in real settings. However, only the smaller version of U-Net model could be applied real time even if only a CPU is at disposal, as it can still process around 45 frames per second while scoring over 0.9 of Intersection over Union metric.

References

[1] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18 (pp. 234-241). Springer International Publishing.

[2] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[3] Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). SegFormer: Simple and efficient design for semantic segmentation with transformers. Advances in Neural Information Processing Systems, 34, 12077-12090.

[4] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Appendix

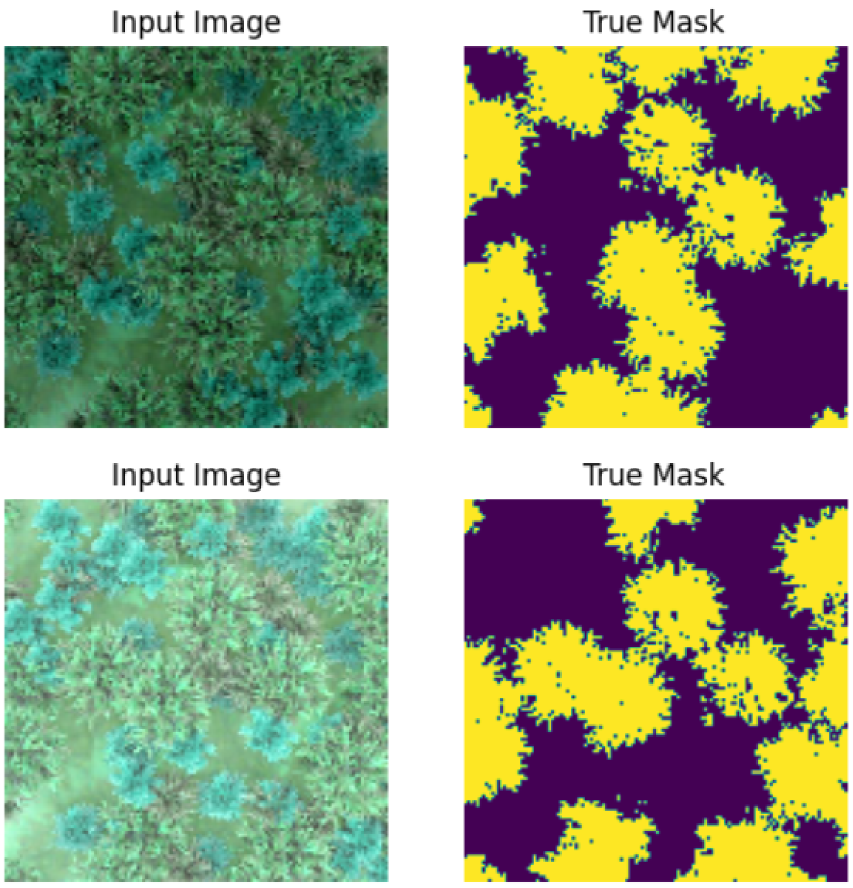


Figure 4: Example of augmentation

Note. In the top we display the original image and mask and their augmented counterparts in the bottom. In this case a combination of diagonal reflection and brightness and contrast adjustment was applied.

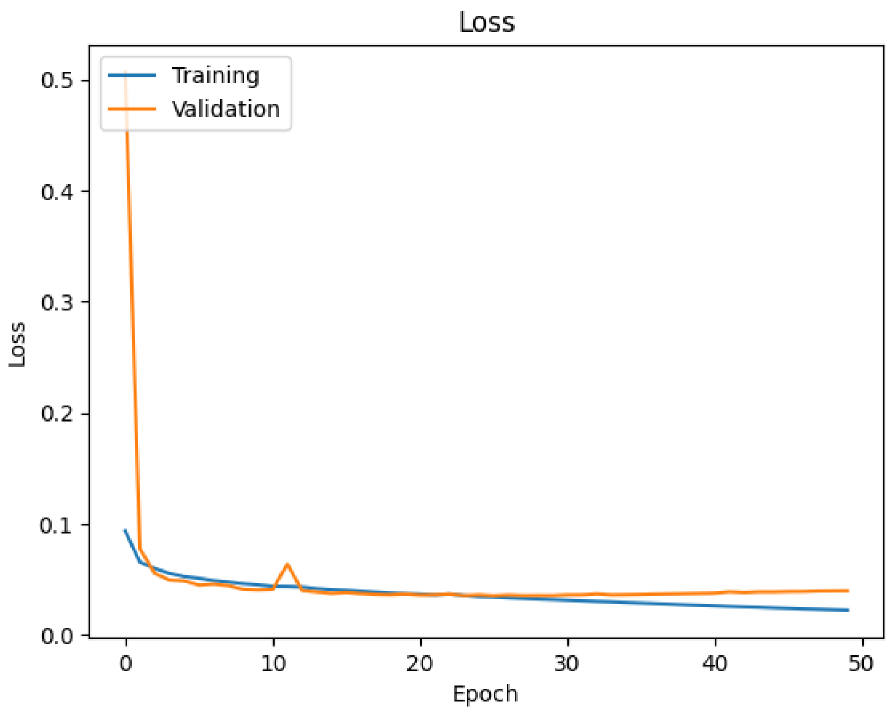


Figure 5: U-Net learning curve

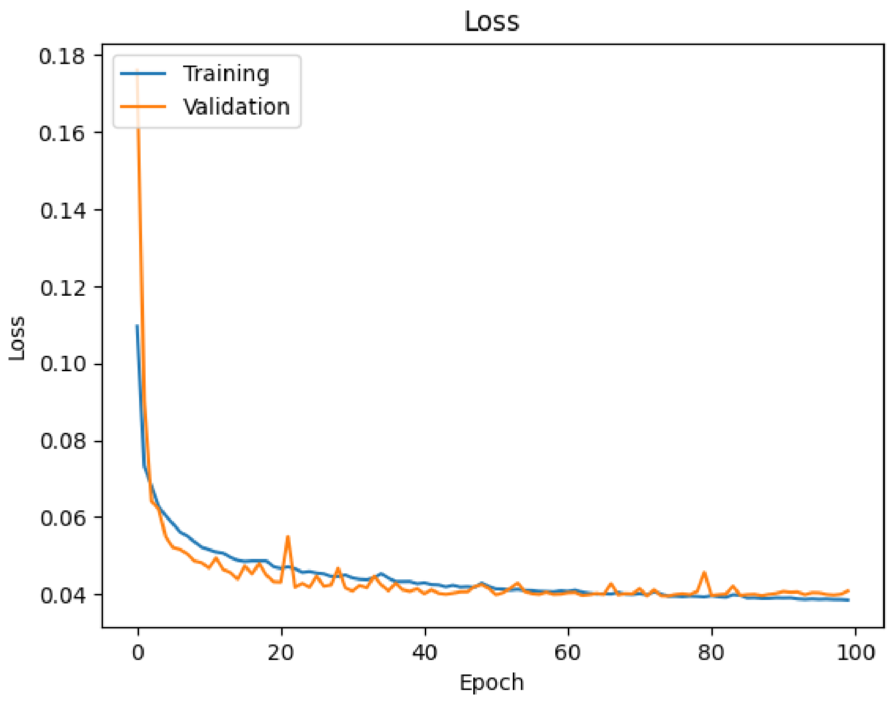


Figure 6: U-Net small learning curve

Filters	Dropout	Learn. rate	Epochs	Loss	Accuracy	IoU	F1-score
8	**0.0**	**0.0050**	**8**	**0.0508**	**0.9395**	**0.8853**	**0.9342**
8	0.0	0.0010	11	0.0529	0.9370	0.8808	0.9314
8	0.0	0.0005	11	0.0577	0.9315	0.8711	0.9255
8	0.2	0.0050	8	0.0543	0.9368	0.8800	0.9296
8	0.2	0.0010	10	0.0579	0.9311	0.8705	0.9257
8	0.2	0.0005	11	0.0605	0.9274	0.8642	0.9226
8	0.4	0.0050	11	0.0561	0.9331	0.8738	0.9273
8	0.4	0.0010	8	0.0624	0.9253	0.8605	0.9203
8	0.4	0.0005	11	0.0678	0.9184	0.8487	0.9136
16	0.0	0.0050	6	0.0477	0.9433	0.8919	0.9380
16	0.0	0.0010	6	0.0494	0.9408	0.8877	0.9359
16	0.0	0.0005	10	0.0494	0.9411	0.8880	0.9355
16	0.2	0.0050	11	0.0468	0.9442	0.8936	0.9393
16	0.2	0.0010	6	0.0507	0.9393	0.8849	0.9342
16	0.2	0.0005	11	0.0515	0.9382	0.8830	0.9334
16	0.4	0.0050	7	0.0499	0.9402	0.8866	0.9354
16	0.4	0.0010	10	0.0522	0.9374	0.8816	0.9323
16	0.4	0.0005	11	0.0548	0.9345	0.8764	0.9291
24	0.0	0.0050	11	0.0419	0.9496	0.9035	0.9451
24	0.0	0.0010	11	0.0413	0.9501	0.9044	0.9459
24	0.0	0.0005	11	0.0437	0.9475	0.8995	0.9425
24	0.2	0.0050	11	0.0445	0.9470	0.8987	0.9420
24	0.2	0.0010	11	0.0420	0.9496	0.9033	0.9446
24	0.2	0.0005	11	0.0467	0.9439	0.8932	0.9395
24	0.4	0.0050	11	0.0443	0.9469	0.8986	0.9423
24	0.4	0.0010	10	0.0462	0.9446	0.8945	0.9400
24	0.4	0.0005	11	0.0493	0.9411	0.8880	0.9358
32	0.0	0.0050	7	0.0473	0.9440	0.8931	0.9380
32	0.0	0.0010	11	0.0395	0.9526	0.9088	0.9480
32	0.0	0.0005	11	0.0431	0.9484	0.9013	0.9437
32	0.2	0.0050	2	0.1887	0.8273	0.6922	0.7716
32	0.2	0.0010	10	0.0429	0.9485	0.9014	0.9438
32	0.2	0.0005	11	0.0434	0.9481	0.9006	0.9427
32	0.4	0.0050	11	0.0426	0.9489	0.9021	0.9442
32	0.4	0.0010	11	0.0426	0.9489	0.9021	0.9439
32	0.4	0.0005	11	0.0461	0.9446	0.8944	0.9395
48	0.0	0.0050	3	0.0555	0.9364	0.8796	0.9303
48	0.0	0.0010	11	0.0395	0.9525	0.9087	0.9481
48	**0.0**	**0.0005**	**11**	**0.0386**	**0.9534**	**0.9104**	**0.9491**
48	0.2	0.0050	3	0.0518	0.9399	0.8859	0.9343
48	0.2	0.0010	11	0.0391	0.9532	0.9098	0.9482
48	0.2	0.0005	11	0.0405	0.9513	0.9066	0.9468
48	0.4	0.0050	11	0.0450	0.9460	0.8971	0.9415
48	0.4	0.0010	11	0.0416	0.9500	0.9041	0.9454
48	0.4	0.0005	11	0.0439	0.9475	0.8996	0.9425

Table 3: U-Net hyper-parameter tuning results

Note. The reported metrics were computed on the validation dataset. The models selected for further training are emphasized with bold font.

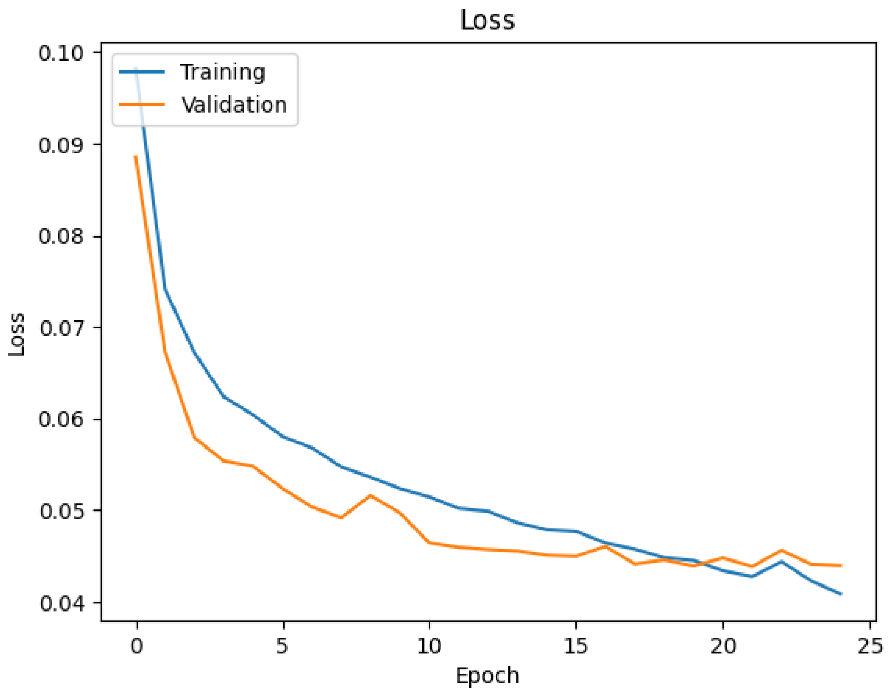


Figure 7: VGG16 U-Net learning curve

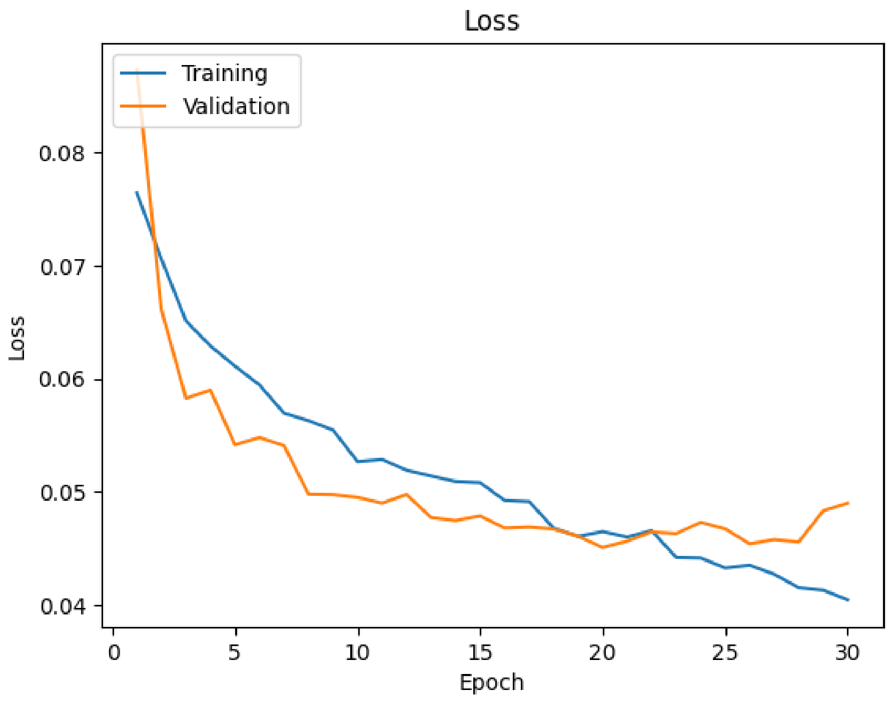


Figure 8: SegFormer learning curve