ITS8030 Homework 2

Talis Tammearu

May 2025

1 Introduction

The chosen task for this homework was to implement a cut forest detector. One of the requirements of the task was to use satellite images from the Sentinel-2 satellite. Another requirement was to use a segmentation model. As part of this homework, the images were fetched through the Sentinel Hub API[1] and manually labeled for the binary segmentation task.

The solution also needed to implement a baseline approach using a convolutional neural network and an attempt to improve the baseline with a foundation model, with the goal of comparing the two methods. To this end, the baseline approach was implemented by training an U-Net [1] decoder for segmentation. In the second part, DINOv2 [4] patch token embeddings were used for segmentation.

Using a segmentation model is appropriate because it gives more information than a simple image classifier. A government agency could be interested in the specific forest allocation that was cut; an image could have more than one allocation of the forest area, and segmentation model would be able to tell which allocations were cut. U-Net architecture was a good choice for the baseline because it is specifically designed for image segmentation and pretrained backbones on Sentinel-2 data already existed. U-Net is a simple model architecture which is important when training with very limited data and it performs better than standard convolutional neural networks. A more complicated architecture with more parameters would probably require more data than one person can label. Similarly, trying to use a foundation model like DINOv2 is appropriate because with very limited data it might be the only way to capture meaningful embeddings from an image, you usually can't expect to train more than a few linear layers.

2 Data gathering and labeling

Data gathering and labeling was by far the most time-consuming part of this homework. First the Sentinel-2 images of forests are needed to label them.

[1] `https://www.sentinel-hub.com`

Identifying cut forests with an untrained eye from 10x10m resolution images seemed like it would take forever, so more data driven approach was taken. First, the list of all forest allocations by county were available to download from the Estonian forest register website[2]. The ids of the forest allocations could be read from a "eraldis.dbf" file with a python script. By observing the requests of their web map, an useful API endpoint was found that returned all cutting on a specific allocation by its' id, along with the year of the cutting. The coordinates of the allocation polygon were also available in the downloaded files. This gave enough information to fetch the initial data. Four 128x128 images at 10x10m resolution were fetched from the SentinelHub API, each image being at different time of the year. 224x224 resolution images could have been a better choice, because it's more common. Only the suitable images around May were chosen for the dataset, because making the models work with images from all seasons was not a goal. May-June was chosen because cut forests are more distinct when the trees are in leaf, especially in the NIR band. The final dataset ended up being 118 images with 80% placed in train split, 10% in validation split and the final 10% in test split. The metadata of the fetched images was also saved so the images could be refetched with different settings if needed.

The images were labeled in CVAT[3] manually, occasionally using the Segment Anything feature. Sometimes, a reference orthophoto from the Estonian geoportal WMS server was needed to help with the labeling. Not all cuttings appeared to be documented, undocumented cuttings had to be identified by eye to improve the quality of the training data. The labeling was done with pixel masks, not polygons. A python script was written to convert COCO format labels into masks in numpy format.

Figure 1: True and false color examples from the dataset.

3 Baseline model

The search for existing segmentation models for satellite images led to the TorchGeo [5] domain library. The library offers pretrained ResNet backbones on satellite images along with easy-to-use training examples. From the options available in TorchGeo, the U-Net architecture was chosen for the baseline segmentation

[2] `https://register.metsad.ee`
[3] `https://www.cvat.ai/`

task. Preliminary tests showed that U-Net achieved higher IoU score over other alternatives like DeepLabV3+[3] on the data collected.

ResNet-50 backbone [2] was used because it was the most capable pretrained backbone available in TorchGeo. Binary cross-entropy was used as the loss function. Cross-entropy loss is commonly used for classification tasks, and since image segmentation can be viewed as a per-pixel classification problem, it is often employed in segmentation tasks as well. Random flipping and random rotations were used for data augmentation.

Three sets of color bands was tested for training the baseline model, see Table 1. True color images always appeared to perform significantly worse than other channel sets, so using true color images wasn't considered further.

Channel Set	**Bands Used**
True Color	B4 (Red), B3 (Green), B2 (Blue)
False Color (Vegetation)	B8 (NIR), B4 (Red), B3 (Green)
All Channels	B1–B12 (incl. B8A)

Table 1: Sentinel-2 Band Combinations

Three models were trained for each of the remaining two channels sets. Jaccard Index (IoU) was used as the primary metric to assess model performance. Unlike accuracy, recall, precision, or F1 score, IoU is insensitive to class imbalance and directly measures overlap quality between the prediction and ground truth segments. Training was done for 40 epoch, because training for longer didn't seem to provide further improvoments on test and validation sets. See the obtained metrics in Table 2. Three models were trained because the metrics seemed to have high variance between training runs. Oddly, the data reveals that one channel set appears to do better on the validation split, the other on test split. Test and validation splits only contained 13 images each, so there could be some bias introduced by the low sample count. Overall, the input channel sets performed similarly enough that it seemed appropriate to continue with only the three channels in the false color set. It would be hard to adapt a 13 channel image to DINOv2, because the model is trained on RGB images. The final IoU achieved by the baseline model was considered to be 0.61 (average IoU of false color image model on test set). Some examples of the predictions can be see on Fig. 2.

Channel Set	**Test IoU**	**Validation IoU**	**Test Accuracy**	**Test Loss**
All Bands	0.5458	0.573	0.9828	0.0610
All Bands	0.5794	0.585	0.9845	0.0539
All Bands	0.5330	0.601	0.9826	0.0576
False Color	0.5717	0.512	0.9841	0.0504
False Color	0.6457	0.546	0.9861	0.0375
False Color	0.6255	0.524	0.9854	0.0415

Table 2: Performance metrics for All Bands and False Color input sets

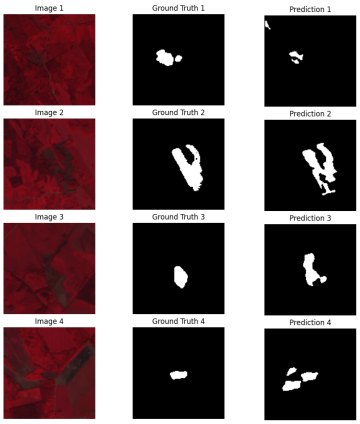

Figure 2: Predictions made by the baseline model along with the ground trurth labels.

4 Foundation model approach

The foundation model approach was based on the DINOv2 model. Dataset masks and images were upscaled from 128x128 to 224x224 to be compatible with the model. Images were upscaled with bilinear filter and the masks were upscaled without filtering.

Considering that the dataset size was only 118 images, at most a couple of linear layers could realistically be trained. Fine-tuning the DINOv2 model itself seemed out of scope. A simple adapter layer on top of the patch token embeddings appeared to be the best approach. A single linear layer appeared to result in the highest IoU metric. The model seemed to perform worse with additional linear layers and ReLU. Applying a per-patch linear layer is quivalent to 1x1 convolution, so 1x1 convolution was used in Torch. The linear layer would still only return a 16x16 mask for 224x224 image - the output needs to be upscaled. A simple bilinear filter appeared to reach the highest IoU score. Upscaling approaches with learnable parameters, like Torch's `ConvTranspose2d` appeared to perform worse.

Other settings were similar to the baseline approach. Binary cross-entropy was used as the loss function. Random flips and rotations were used for data augmentation. The DINOv2 ViT-L/14 model was used for evaluation. Training duration was set at 20 epoch as further training didn't improve IoU on test and validation splits.

Around 0.52 IoU score was reached on the test split and 0.58 on the validation split. Example of the predicted masks can be seen on Fig. 3.

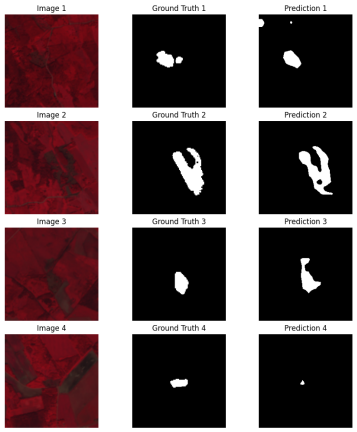

Figure 3: Predictions made by the model based on DINOv2.

5 Conclusions

More traditional deep learning approaches got better results on this task. DINOv2 foundation model was not trained on satellite images, so it's quite likely that the model was unable to understand the visual features of the images. An approach that combines DINOv2 patch embeddings with ResNet-50 embeddings might get better results, but wasn't implemented in this work. Labeling more data could help the models to learn more intricate patterns and improve the validation metrics. With more data it could also be an option to fine-tune the DINOv2 model to better understand features of satellite images.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. eprint: `arXiv:1505.04597`.

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[3] Liang-Chieh Chen et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *arXiv preprint arXiv:1802.02611* (2018).

[4] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2023.

[5] Adam J. Stewart et al. "TorchGeo: Deep Learning With Geospatial Data". In: *ACM Transactions on Spatial Algorithms and Systems* (Dec. 2024). DOI: `10.1145/3707459`. URL: `https://doi.org/10.1145/3707459`.