Road segmentation from satellite images

Ondrej Kováč (xkovac57) Jakub Július Šmýkal (xsmyka01)
Jan Svoboda (xsvobo2b)

December 23, 2024

1 Introduction

This work focuses on road segmentation from satellite images. It contains the process of implementation, training and testing of various convolutional neural networks, which are able to identify and create masks for roads in satellite images. Road segmentation can be useful in many areas, such as urban planning, disaster response, autonomous navigation and so on.

2 Existing solutions

As this is an image segmentation task, it was obvious to us, that we would be using a convolutional neural network. We looked at 3 different papers [3][4][5] focusing on this specific issue and found, that they use very similar approaches. Namely, all of them used some variation of the U-Net architecture.

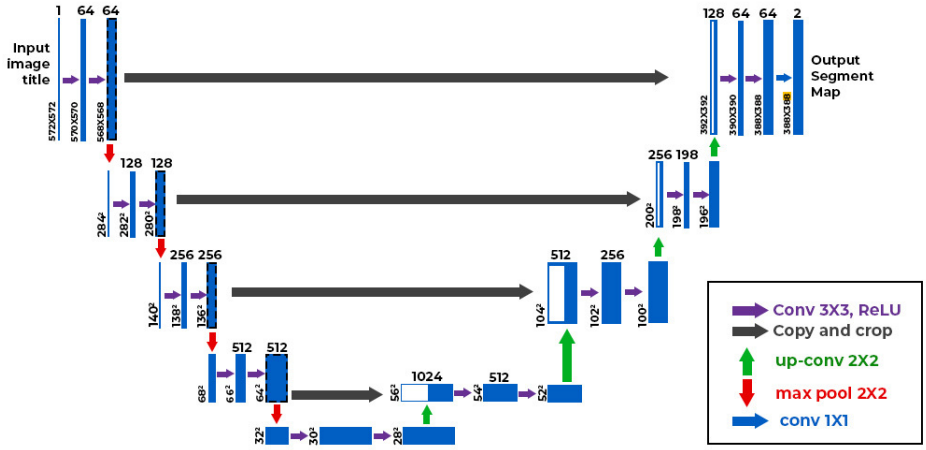

Figure 1: U-Net architecture example. Sourced from [2]

U-Net is an architecture specifically designed for image segmentation tasks and consists of two main parts:

- Encoder - captures context and features from the input image through a series of convolutional and pooling layers, reducing the spatial dimensions while increasing feature depth.

- Decoder - Restores spatial resolution through up-convolutions (transposed convolutions) and combines these with high-resolution features from the encoder using skip connections, which help preserve spatial details.

The papers used different combinations of loss functions (Dice Loss, BCE...) and optimizers. All of them, however, had one evaluation metric in common, and that was Intersection over Union (IoU).

$$\text{IoU} = \frac{|\text{Prediction} \cap \text{Ground Truth}|}{|\text{Prediction} \cup \text{Ground Truth}|}$$

If we compare their results using the mean IoU metric over the testing dataset we get the following results:

- Paper [3] tested 5 different U-Net architectures:
  - U-Net - 0.599
  - ResUNet - 0.600
  - AttUNet - 0.616
  - C-UNet - 0.635
  - Their Proposed U-Net model - 0.620
- Paper [4] reached an IoU of 0.73
- Paper [5] tested a U-Net with 4 different backbones:
  - ResNet18 - 0.711
  - ResNet34 - 0.717
  - ResNet50 - 0.735
  - ResNet152 - 0.732

These values, of course, cannot be compared directly, as the models were not trained and tested on the same data. The results do however give us an idea about the difficulty of this task. These scores, if taken at face value, actually seem quite low. We will get into the hypothesized reasons why IoU scores in this particular problem are relatively low when we explain our own model.

3 Our Implementation

3.1 Goal

As the goal of our project, we decided to test multiple different architectures and learning methods and compare their performance. Then we selected the best-performing combination and properly trained it to see if it could compare with existing solutions mentioned in the previous chapter.

3.2 Environment

We decided to create our project[1] in Python and use the **Pytorch** library for our neural network. We then tested models from **Python Segmentation Models** library [6].

3.3 Dataset

We decided to use the **DeepGlobe Road Extraction Dataset** [1]. This dataset contains more than 6000 training images with corresponding ground truth masks.

3.4 Data augmentation

We used online augmentations during training. Each time an image from the training set was augmented using the **Albumentations** library. This meant that the model was trained and evaluated on a slightly different dataset each epoch.

These augmentations modify the resolution and the color of images. Random part with user-selected resolution is cut out of the original image and randomly rotated. This newly created image has then modified color tone. There are also some additional augmentations, which are optional and can be used to create even more diverse training data. These are addition of blur, randomly adjusting brightness, contrast, saturation and hue and transformation of the image to grayscale.

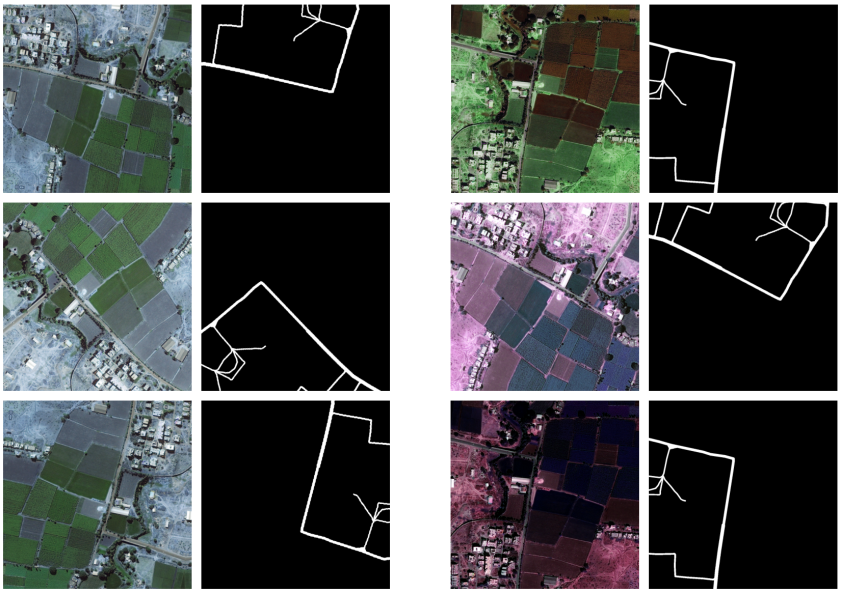

Figure 2: Examples of image and its mask after multiple augmentations. Images on the right side are created with more extensive augmentations enabled.

4 Training and testing

For the training, we used Dice's loss function. For the accuracy evaluation, we used the intersection of unions metrics (IoU). We found out that IoU was sometimes low in

[1] GitHub Repository

comparison to the quality of the graphical results. When comparing the images, we observed that the quality of the dataset was not very good, as the visually present paths were often not labeled in the dataset, as can be seen in figure 3. This also caused problems in the training because the model was penalized for correctly segmented paths. Another problem is that paths can be very thin, so even a small deviation can cause a large drop in the IoU.

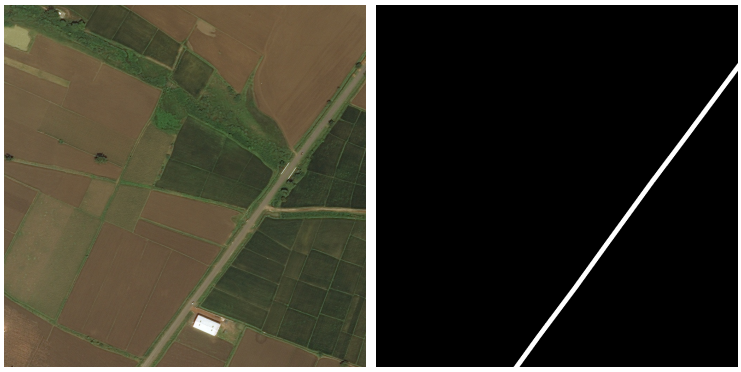

Figure 3: Missing path example.

4.1 Encoder testing

The encoders resnet101, efficientnet-b4, and efficientnet-b5 were tested on 720 training and 180 validation images. We used pretrained weights. Training was performed on the UNET architecture in 100 epochs and for each epoch, 3 random augmentations of size 256x256 were created from each image and used for training. Finally, 100 images were tested in full size (scaled but without cropping). A comparison of the different encoders is shown in the table 2.

Encoder	Training Loss	Training IoU	Testing IoU
resnet101	0.360	0.486	0.457
efficientnet-b4	0.258	0.606	0.541
efficientnet-b5	0.258	0.607	0.554

Table 1: Encoder testing results.

It turned out that eficientenet-b5 performed the best. When comparing the graphical outputs, eficientnet-b5 showed very good results so no other encoders were tested and this encoder was used in further experiments. Graphical results of this test can be seen in figure 4.

4.2 Architecture testing

For testing, we decided to test all architectures available from the Python Segmentation Models library except LinkNet in the same way as the encoders. The results can be seen in the table 2.

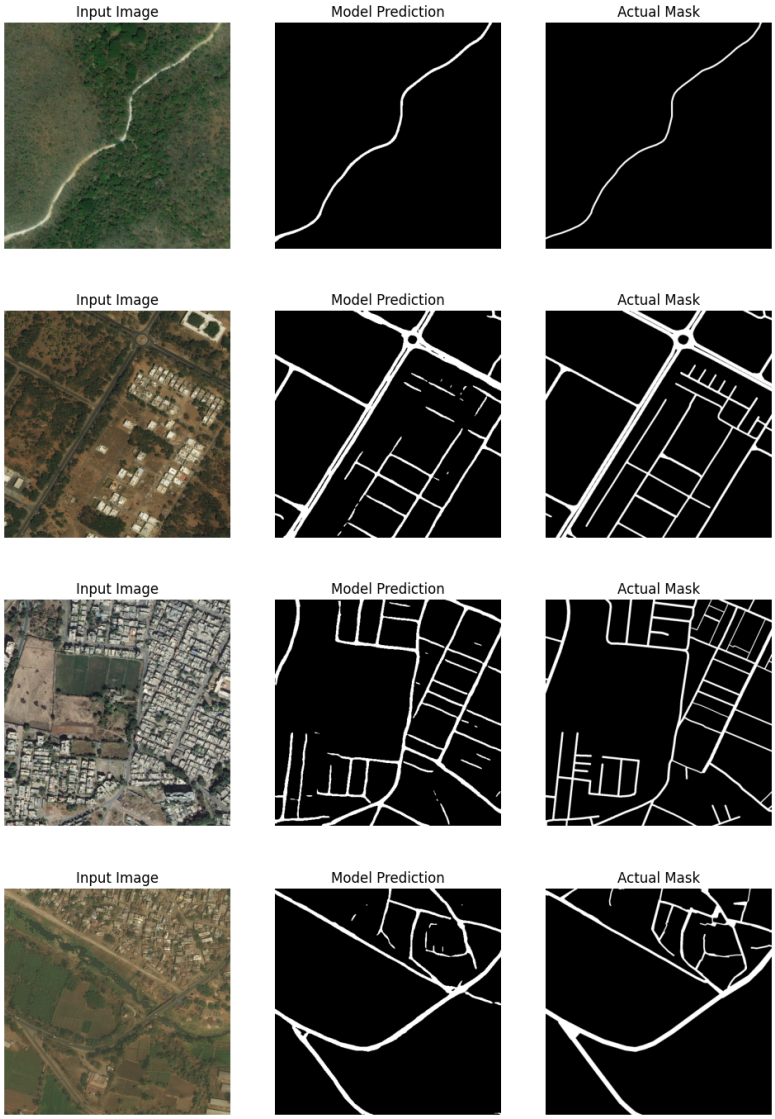

Figure 4: Encoder testing efficientnet-b5 results.

Model	Training Loss	Training IoU	Testing IoU
FPN	0.249	0.621	0.529
PSPNet	0.309	0.554	0.431
PAN	0.259	0.613	0.507
UNET	0.258	0.607	0.554

Table 2: Architecture testing results.

In the table, it can be seen that FPN has the best training parameters, but it has worse results on the testing images. The visual result was also blurrier and less detailed. In the test images, UNET achieved the best results, but the PAN model converged faster in the first few epochs. The visual results were also promising for PAN after a few epochs. In the final results, the PAN has worse results than UNET. It can be caused by different resistance to errors in the training dataset. It's also good to mention that PAN is much slower than UNET.

4.3 Final testing

For the final testing, we used the UNET architecture with the efficientnet-b5 encoder. We used 2480 images for training, 620 for validation and we trained the model for 150 epochs. The training curve can be seen on figure 5. Only one augmentation for each image per epoch was used. The results can be seen in the table 3.

Model	UNET
Encoder	efficientnet-b5
Training Loss	0.2705
Training IoU	0.5905
Testing IoU	0.6463

Table 3: Final model parameters.

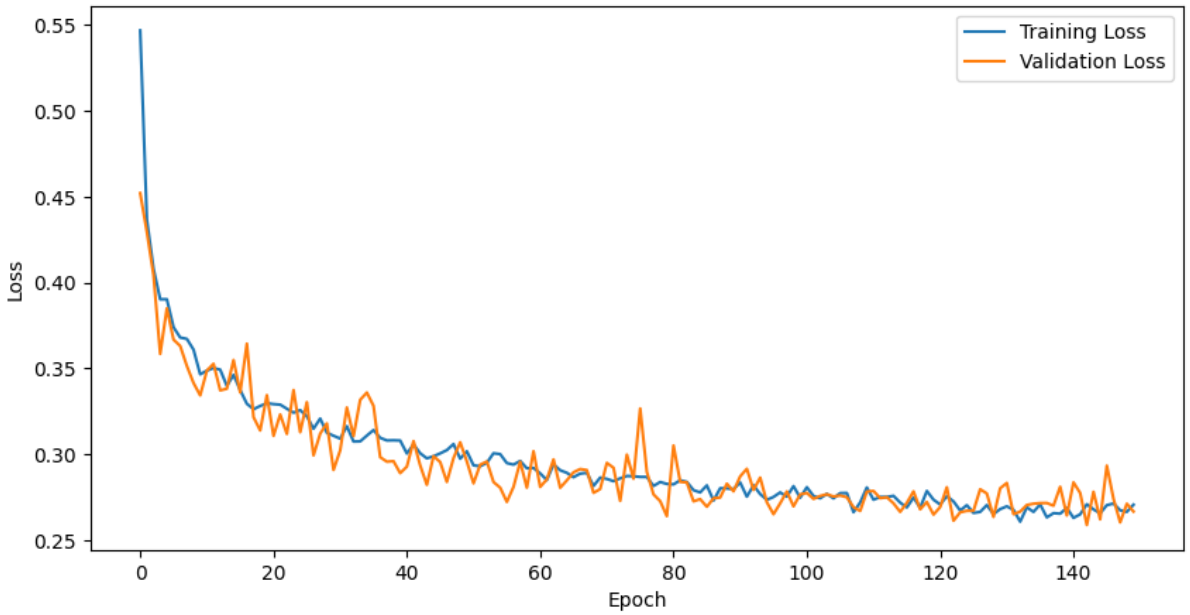

Figure 5: Training curve for the final training.

The graphical results can be seen in 6. According to the graphical results, the model performs well in path recognition and successfully ignores other structures that might resemble paths. However, in the images, it can also be seen that for thin paths, the model has trouble recognizing what is a path and what is not. We believe this is primarily due to the problematic dataset, where some paths, especially the less frequent ones, are not labeled in the dataset at all.

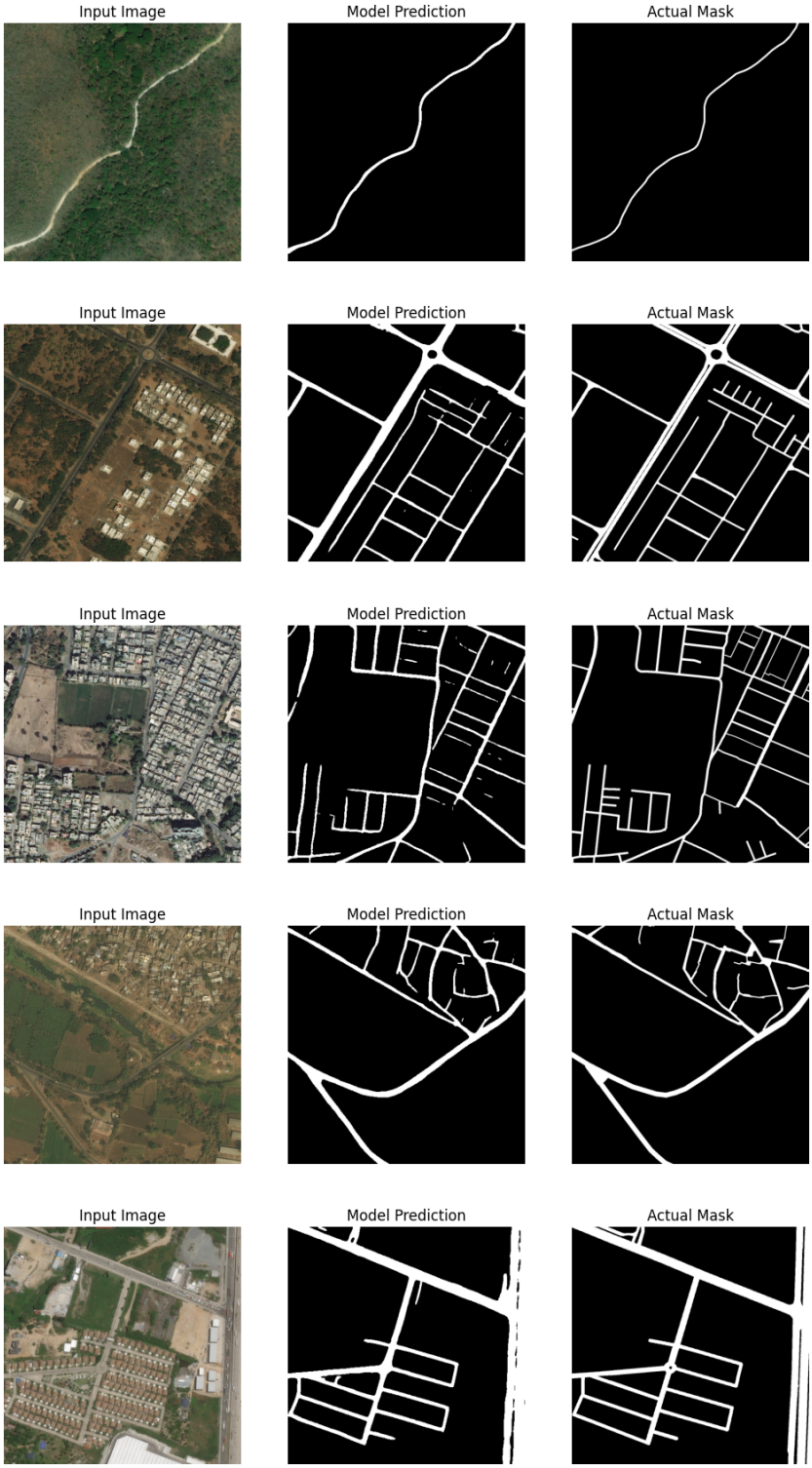

Figure 6: Final testing results.

5 Conclusion

We have successfully trained a convolutional neural network for road segmentation that is able to create masks for roads in satellite images with accuracy comparable to existing models. Despite the relatively low quantitative accuracy, which can mostly be attributed to the low quality of the dataset, we are pleased with the qualitative results. Future improvements could include refining the dataset and removing samples with poor ground truth masks, which would undoubtedly make the model more accurate.

References

[1] Balraj Ashwath. *DeepGlobe Road Extraction Dataset*. 2020. URL: https://www.kaggle.com/datasets/balraj98/deepglobe-road-extraction-dataset/data.

[2] GeeksForGeeks. *U-Net Architecture Explained*. 2023. URL: https://www.geeksforgeeks.org/u-net-architecture-explained/.

[3] Preetpal Kaur Buttar Manoj Kumar Sachan Vidhi Chaudhary. "Satellite imagery analysis for road segmentation using U-Net architecture". In: *The Journal of Supercomputing* (2022). URL: https://link.springer.com/article/10.1007/s11227-022-04379-6.

[4] Nithish Murugesan. "Satellite Imagery Road Segmentation". In: *Medium* (2022). URL: https://medium.com/@nithishmailme/satellite-imagery-road-segmentation-ad2964dc3812.

[5] Ozan Ozturk et al. "Improving Road Segmentation by Combining Satellite Images and LiDAR Data with a Feature-Wise Fusion Strategy". In: *Applied Sciences* 13.10 (2023). ISSN: 2076-3417. DOI: 10.3390/app13106161. URL: https://www.mdpi.com/2076-3417/13/10/6161.

[6] Pavel Yakubovskiy. *Segmentation Models*. 2018. URL: https://github.com/qubvel/segmentation_models/tree/master.