Documentation

Vision Transformers for Image Segmentation

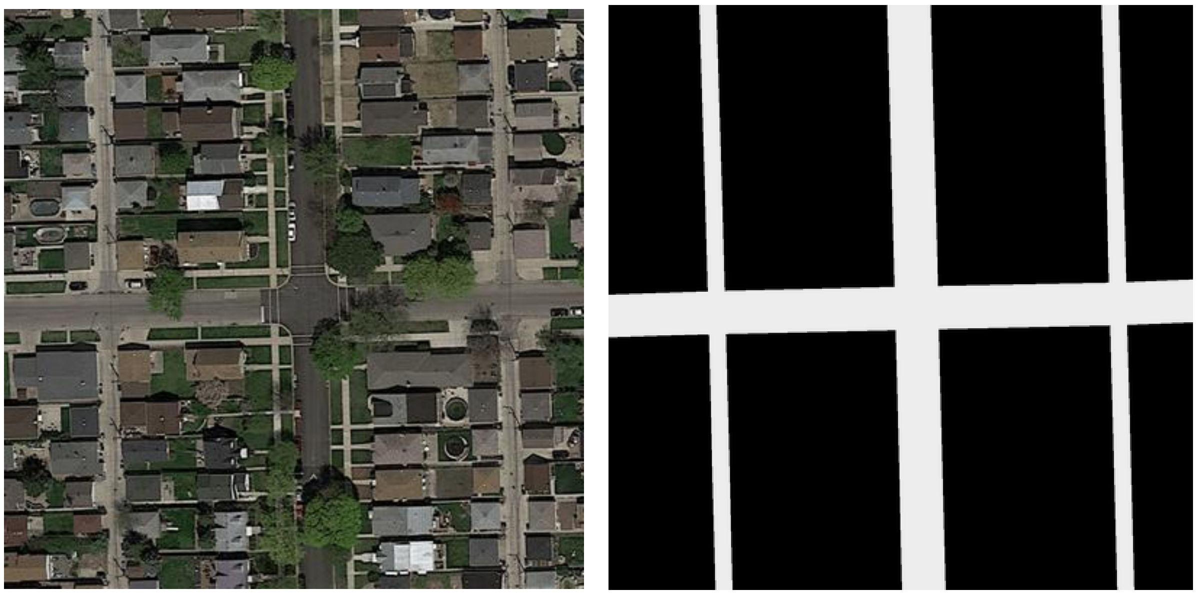
Introduction

Our chosen topic is Vision Transformers (ViT) for Image Segmentation. Vision Transformers are specifically designed for computer vision tasks, leveraging the transformer architecture originally developed for natural language processing.

Compared to convolutional neural networks (CNNs), which were predominantly used for image segmentation in the past, Vision Transformers offer a key advantage: they utilize global context. While CNNs are limited to focusing on local features, Vision Transformers use self-attention mechanisms to capture relationships across the entire image. This ability to use global information makes Vision Transformers powerful for image segmentation tasks.

Dataset

For our dataset, we selected a road layout segmentation dataset from Kaggle. These are top-down-view pictures of maps with their masks to train the neural network.



The dataset contains 6226 images for training, with additional for validation and testing.

Methods of training

For the baseline models, we used convolutional neural networks. The first one is a U-Net architecture. U-Net fully convolutional neural network, originally developed for medical imaging tasks. This architecture uses an encoder and a decoder and extracts the information through a bottleneck. It also uses skip-connections, which means passing input into a deeper layer, skipping intermediate layers.

The other one is a DeeplabV3 model from torchvision. DeeplabV3 segmentation architecture is developed by Google researchers, which are particularly effective for image processing tasks. It uses dilated convolution, which can increase the receptive field without increasing the number of parameters. It has accurate results, but it is slow due to its size and computational requirements.

For the Vision Transformer model, we used a Swin-UNET model, which combines the advantages of both the Swin and U-Net. The Swin Transformer architecture offers hierarchical feature representation and a window attention mechanism while the U-Net architecture gives the encoder-decoder structure and skip-connections.

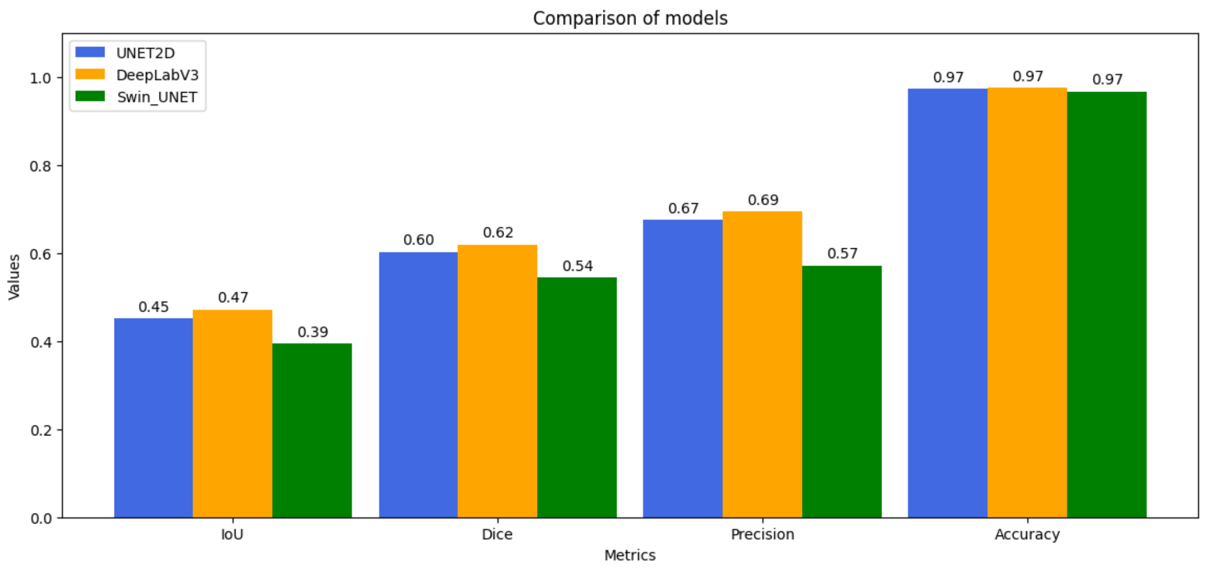
We followed the model trainings on Weights and Biases. We optimized the hyperparameters manually on low epoch, and then used the parameters that got the best results, and trained the models further.

For our loss function we used a combination of binary cross entropy and Dice. The BCE loss considers all pixels with the same importance, regardless, no matter how common that class is in the images. Since background pixels are much more abundant, they dominated the averaging and the model performed better at detecting them, while the detection of pixels associated with paths became much less accurate. The problem was solved by adding a Dice loss to the binary cross-entropy (BCE). Dice loss measures the overlap between predicted and real masks. Dice loss is less affected by class imbalance than, BCE loss, so it can handle disproportionalities better. During the learning phase of segmentation models, we used the linear combination of the two losses. By combining the two loss functions, the models can take advantage of strengths of both, resulting in a more accurate and robust segmentation capability.

Evaluation

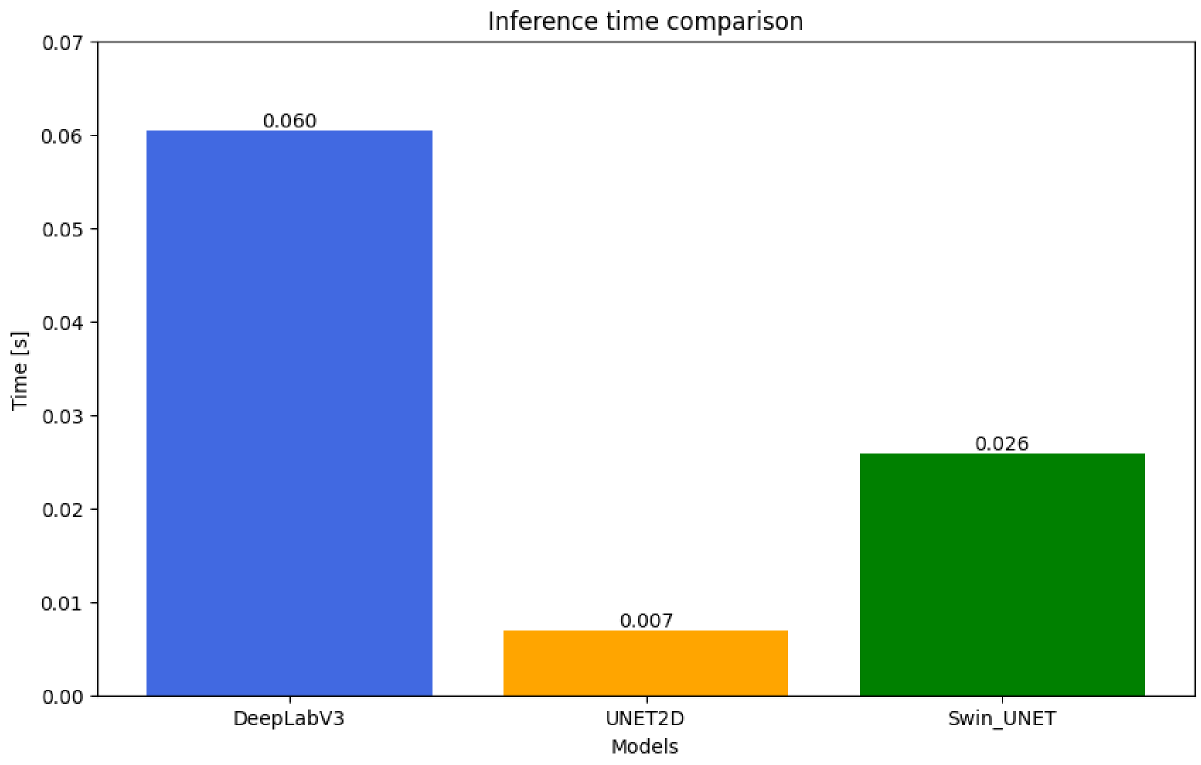
To evaluate the models, we used multiple different key metrics. Since in most images, the pixels classified as not road far exceed the pixels classified as road, accuracy is not too effective to describe the difference between model's performance. A better variable is Intersection over Union (IoU), which means the intersections of the prediction and the mask divided by the union. Another useful evaluation criteria which we used is DICE coefficient, which measures the overlap between the predicted value and the truth. Precision measures the rate of correct road classifications.

On the key metrics, the models achieved these percentages:



Based on this, the best performing model was the DeeplabV3, followed by the UNet2D, while the Swin-UNET model had the worst performance.

We also compared the models based on inference time. Here is the comparison:



The results showed that the U-Net model is much faster than the DeepLabV3, and the Swin-UNET model is also faster, because of the DeepLabV3's high parameter count.

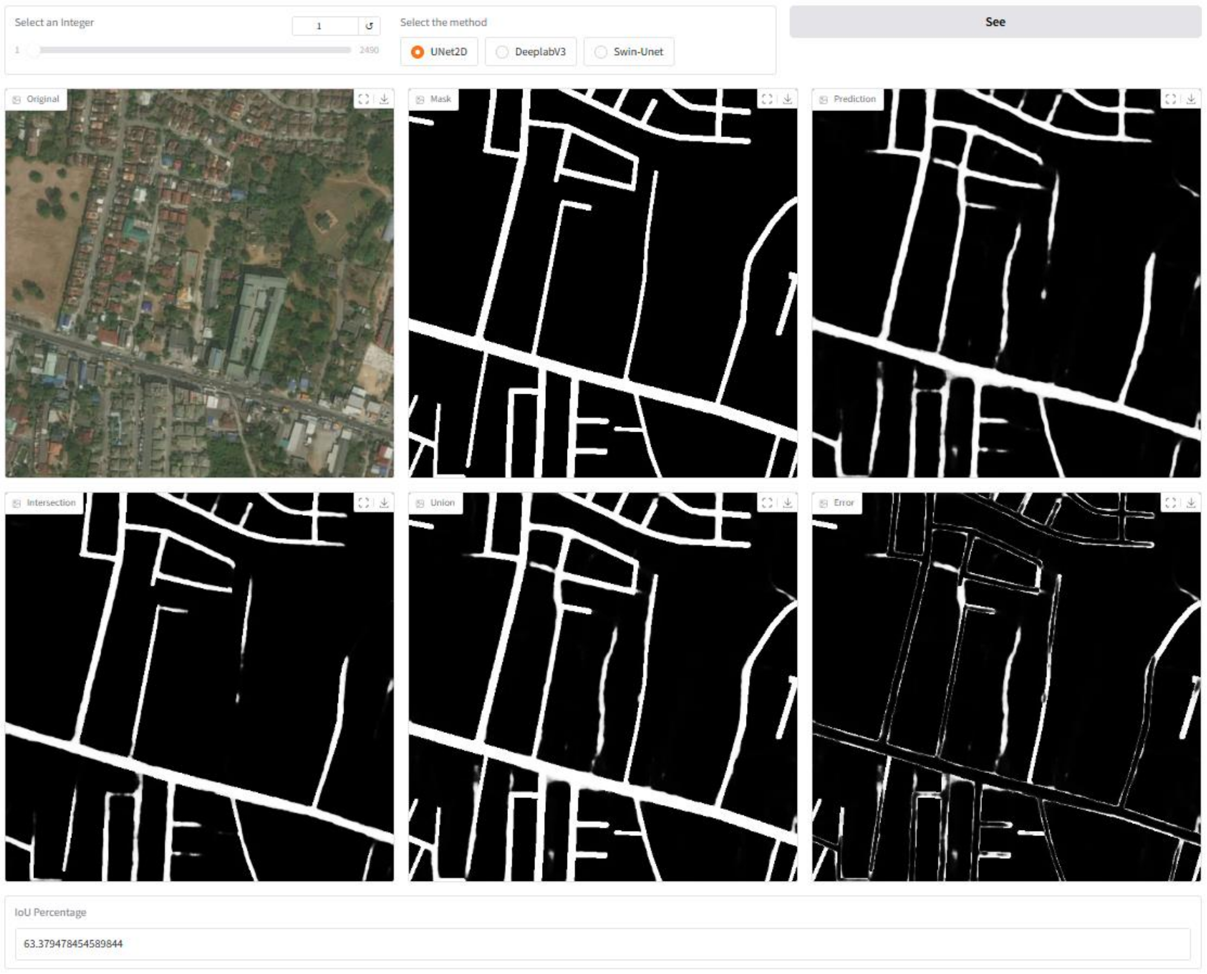
Comparing the models on ease of implementation, the DeeplabV3 was the easiest, since we used a pre implemented model. In the training phase, it took the longest (for one epoch), because of the high number of parameters, but converged quite quickly.

The U-Net architecture was also easy to implement, because it is a well-known architecture. The training was fast, and it also converged quickly, so the learning phase was the shortest.

The Swin-UNET model was the hardest to implement, and most complicated to understand, the model implementation was taken from the referenced GitHub repository. The model's hyperparameters are locked, and they were dependent on each other, meaning with some set ups, it did not learn at all. This model converged slowly, but it was resistant to overfitting.

Visualization

We made a few examples of each model with the original image, the mask and the model's prediction for the image. We also visualised and compared the models' performances in the Jupiter notebook. Additionally, we created a frontend UI using Gradio.



Select an Integer: 1

Select the method: [✓ UNet2D] DeeplabV3 Swin-Unet

IoU Percentage: 63.379478454589844

On the UI, you can choose a random image from the dataset, and model to evaluate. It shows the models prediction to the image, the intersection and the union of the models. It also shows the differences between the mask and the prediction (error of the prediction). It also tells the IoU of given picture with the selected model.

Conclusion

With this project, we became more familiar with the Vision Transformer architecture, while comparing them to CNN solutions which were our baseline models.

From the results Vision Transformers can compete with regular CNN methods in Image Segmentation tasks. However, the Vision Transformer model based performed still performed worse than the CNN-based baselines.

Our dataset was quite simple and small sample size. In these types of datasets, CNN-based solutions probably perform better, then ViT approaches. But with larger datasets and more complex Image Segmentation tasks, ViT can have a better performance and scalability compared to CNN-s.

LLMs

We used Large Language Models for different parts of the project. The IDE has an integration for Copilot which we used for code completion. We also used LLMs for debugging and code generation. In the documentation, we used it for inspiration and text correction.

References

Road layout image segmentation: https://www.kaggle.com/datasets/balraj98/deepglobe-road-extraction-dataset/data

WandB: https://wandb.ai/site/

U-Net: https://arxiv.org/abs/1505.04597

DeeplabV3: https://pytorch.org/hub/pytorch_vision_deeplabv3_resnet101/

Swin-Unet: https://arxiv.org/abs/2105.05537

Swin-Unet model: https://github.com/HuCaoFighting/Swin-Unet/tree/main

Loss function: https://arxiv.org/abs/2209.00729

Gradio: https://www.gradio.app/