Applied Segmentation Models for Identifying Deforestation Drivers

Simon Vedaa[1], Khalil Ibrahim[1], Safiya Mahamood[1], and [2]

[1]University of Bergen

Abstract

We evaluate several state-of-the-art semantic-segmentation architectures on a deforestation-driver mapping challenge hosted by Solafune. Our pipeline applies standard four-channel tensor conversion of annotated polygons, extensive geometric and photometric augmentations, and a minimum-area filter in post-processing. We compare UNet, DeepLabV3+, SegFormer, Vision Transformer (ViT), TransUNet, and two ensemble variants in terms of segmentation accuracy (pixel-wise F1) and computational cost. DeepLabV3+ trained on all 12 Sentinel-2 bands achieved the highest validation F1 (0.7367) and earned 0.5851 and 0.5624 on the public and private leaderboards, respectively. Contrary to expectations, larger architectures did not outperform smaller ones under our resource constraints. We analyze this behavior, attribute it to overfitting and limited fine-tuning of huge backbones, and discuss how more extensive tuning and data could better realize their potential.

Introduction

In this paper we are working on the task of identifying deforestation drivers. This is part of a Solafune ML competition[1] where the goal is to classify and segment drivers of deforestation in satellite imagery.

Several architectures have been developed for image segmentation in different fields, such as UNet (Ronneberger, Fischer, and Brox 2015) and TransUNet (Chen et al. 2021) for medical image segmentation, or the Segformer (Xie et al. 2021) for general purpose segmentation.

We have several goals we want to achieve. First, achieve the best possible performance on the competition dataset. Second, compare the different implementations in terms of performance vs computational cost. Lastly, to reproduce results reported in the original architecture papers, which we have implemented.

Our contributions consist of applying different segmentation model architectures on a deforestation segmentation task, and comparing their performance.

Methods

Our entire pipeline is based upon a GitHub repository made by motokimura.[2] We have made a couple of modifications to the training and evaluation pipeline, but the pre- and post-processing steps remain mainly unchanged.

Pre-processing

The competition data comes in the form of annotated polygons in a json file. We convert those into tensors of 4 channels. We apply standard image augmentations such as flipping, scaling and rotation. Additionally we apply random cropping reducing the input image by half. The cropping is not applied for the Vision Transformer (ViT) model. Images are also normalized using mean and standard deviation calculated from training images. Normalization values depend on the number of input channels, i.e. using only RGB or all channels.

Post-processing

We applied a score threshold of 0.5 to binarize the predicted masks. Additionally, we discard masks smaller than 10,000 pixels during inference. The idea is that removing small segmentations reduces the number of false positive predictions.

Model architectures

We have applied the following model architectures; UNet, DeepLabV3+, Vision Transformer (ViT), Segformer, and TransUNet. We used the Pytorch Segmentation Models library (Iakubovskii 2019) to implement UNet, DeepLabV3+, and Segformer, while ViT and TransUNet are implemented following their respective papers and source code.

Vision Transformer (ViT)

The Vision Transformer (ViT) model treats an image as a sequence of fixed-sized patches and processes them with a standard Transformer encoder. It relies on self-attention instead of convolutional inductive biases to learn long-range dependencies.

[1] Competition website

[2] Baseline pipeline by motokimura

Proceedings of the 7th Northern Lights Deep Learning Conference (NLDL), PMLR 1, 2026.

ViT Architecture The Vision Transformer breaks an image into fixed size patches, linearly embeds each patch (plus a learnable class token and positional embeddings) into vectors. The result of this is then fed as sequence through standard Transformer encoder layers that contains multi-head self-attention followed by feed-forward networks, using the final class token as representation for prediction. Its architecture can be seen in Figure 1

ViT Implementation In our ViT implementation we started from the torchvision ViT-B/16 model pretrained on ImageNet (Dosovitskiy et al. 2021) swapped its first layer to accept all 12 Sentinel-2 bands, resized its built-in positional embeddings to our 1024×1024 image grid, and replaced the classification head with a lightweight segmentation head for four land-use classes. The transformer backbone remained frozen, and only the new segmentation head was trained on our data.

Why ViT? Based on (Dosovitskiy et al. 2021) the ViT applies a pure Transformer to image patches and, with large-scale pre-training, matches or exceeds CNNs on vision benchmarks. Since our task requires capturing long-range, multispectral context in high-resolution satellite imagery, we wanted to see if ViT could similarly improve segmentation performance.

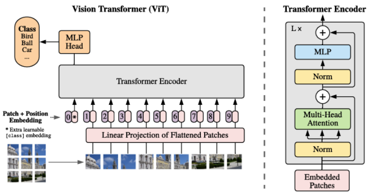

Figure 1. Vision Transformer Architecture (Dosovitskiy et al. 2021)

Segformer

SegFormer (Xie et al. 2021) is a transformer-based architecture designed for efficient semantic segmentation. It combines the strengths of hierarchical representations from convolutional networks with the global context modeling of transformers. In our project, we included SegFormer as one of the core models to evaluate its ability to identify deforestation drivers in satellite imagery.

Segformer architecture SegFormer consists of two main components (Figure 2): the Mix Transformer (MiT) encoder and a lightweight MLP-based decoder. The encoder is optimized for visual tasks, using overlapping patch embeddings and a hierarchical structure to effectively capture both local and global image features. Unlike traditional Vision Transformers, SegFormer replaces explicit positional encodings with Mix-FFN modules, improving robustness to varying input resolutions.

The decoder is composed entirely of Multi-Layer Perceptrons (MLPs), which aggregate multi-scale features from the encoder. This design keeps the decoder lightweight while maintaining strong segmentation performance.

Segformer implementation We used the implementation of SegFormer provided by the *segmentation_models* library (Iakubovskii 2019), which integrates smoothly with PyTorch and supports modular experimentation. This allowed us to quickly prototype and evaluate different model variants under our unified training and evaluation pipeline.

Why SegFormer? We selected the SegFormer-B5 variant, which is the most powerful configuration of the architecture. Its deep encoder is particularly effective at capturing both fine-grained and large-scale features, making it well suited for complex segmentation tasks like deforestation mapping. In addition, SegFormer has demonstrated strong benchmark results on datasets such as Cityscapes and ADE20K, indicating reliable generalization to a variety of segmentation domains.

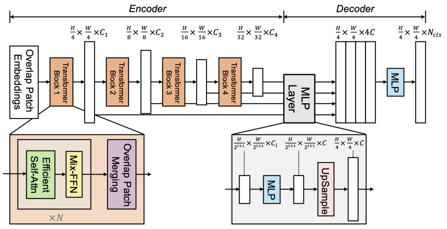
Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

Figure 2. Segformer architecture (Xie et al. 2021)

TransUNet

TransUNet Architecture TransUNet is very similar to its predecessor UNet. It consists of an encoder and decoder architecture, where the main difference is the introduction of a transformer in the encoder as seen in Figure 3. The decoder block called CUP, short for Cascaded Upsampler, consists of multiple upsampling blocks, which are made up of a 2x bilinear upsampler followed by two convolutional blocks. The decoder also uses skip connections from the CNN encoder, passing them into the first convolutional block at each upsampling stage.

TransUNet Implementation In our implementation we use ResNet50-VisionTransformer for the hybrid encoder, using pre-trained weights loaded from the `timm` library. We implement the base version of TransUNet as they do in (Chen et al. 2021).

Because our inputs have three or more channels, we replaced the ResNet encoder's first convolutional layer; the rest of the hybrid encoder remained unchanged.

Why TransUNet? According to (Chen et al. 2021), TransUNet is an improvement to UNet for the task of medical image segmentation. Since we use UNet as one of our baseline models, we were interested to see if we could get similar results for our task.

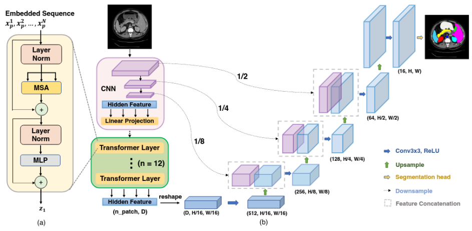

Figure 3. TransUNet architecture (Chen et al. 2021)

Ensemble models

We create two ensemble models, one with all models called `ensemble1` and one without TransUNet called `ensemble2`. Ensemble models average the output logits of all its models. As shown in Results, TransUNet without post-processing performs significantly worse than the other models.

Training and evaluation

Hyperparameters

Across all models, we use a learning rate of 1e-4 and a weight decay of 1e-2, which we found to balance convergence speed and regularization. Training is parallelized using 12 workers to optimize data loading efficiency.

Loss and Metric

The loss function is the sum of Dice loss and Soft Binary Cross-Entropy with a smoothing factor of 0.

We use the pixel-based F1 score as the evaluation metric, in line with the competition rules. It balances precision and recall based on the overlap between predicted and ground truth masks, computed per class and averaged across classes.

Batch size and gradient accumulation

As some of the models are quite large, and we have limited resources, we decided to use batch gradient accumulation. Instead of using larger batches, we use smaller k batches and accumulate the gradients of N total batches before the backward pass. The effective batch size then becomes $k*N$. All models are trained on an effective batch size of either 15 or 16. We used pytorch lightning's built in batch gradient accumulation. Batch sizes and accumulation steps are tuned based on each model's computational cost and memory footprint.

Learning rate scheduler

We use an AdamW optimizer with a cosine-decay schedule. The learning rate starts at our base value and smoothly decays to zero over the full training run, with no explicit warmup period. The cosine curve ensures that the LR decreases gently at first and then more rapidly toward the end of training.

Channel input

Each model is trained in two variants, one ingesting all 12 Sentinel-2 spectral bands and one using only the standard RGB channels, so we can measure the benefit of the extra multispectral information.

Frozen start

We freeze the Transformer encoder for the first five epochs, while TransUnet uses 15 epochs updating only the new segmentation head before unfreezing the backbone for joint fine-tuning.

Training process

We train each model for 200 epochs, evaluating on the validation set every five epochs. We use a 80/20 split for training and validation sets. The final version of the model we keep, is the one that achieves the highest f1 score throughout training.

Model selection

Once every model is finished training, we run them through our post-processing step, and calculate their validation score. The model with the highest score is then chosen and used to generate the final predictions for the test set, i.e. the competition submission.

Results

Effect of adding minimum area

Adding a minimum-area filter on the predicted masks, substantially improves model performance, as seen in Table 1. Notably, this more than doubled TransUNet's F1 score.

Effect of channels

Models trained on all channels perform marginally better than their RGB-only counterparts. When looking at Figure 4 and Figure 5, both produce similar segmentations, but the RGB-only models

Model	Min area = 0	Min area = 10k
unet_rgb	0.5961	0.6917
deeplab_rgb	0.6289	0.7159
segformer_rgb	0.6174	0.7029
vit_seg_rgb	0.6652	0.7200
transunet_rgb	0.2089	0.6514
ensemble1_rgb	**0.6727**	0.7182
ensemble2_rgb	0.6725	0.7180
unet_full	0.6303	0.6906
deeplab_full	0.6520	**0.7367**
segformer_full	0.6302	0.7048
vit_seg_full	0.6098	0.7072
transunet_full	0.2456	0.5915
ensemble1_full	0.6706	0.7335
ensemble2_full	0.6698	0.7327

Table 1. Validation f1 scores with and without Minimum Area of 10k(pixels)

generate noticeably more false positives (especially in the final row).

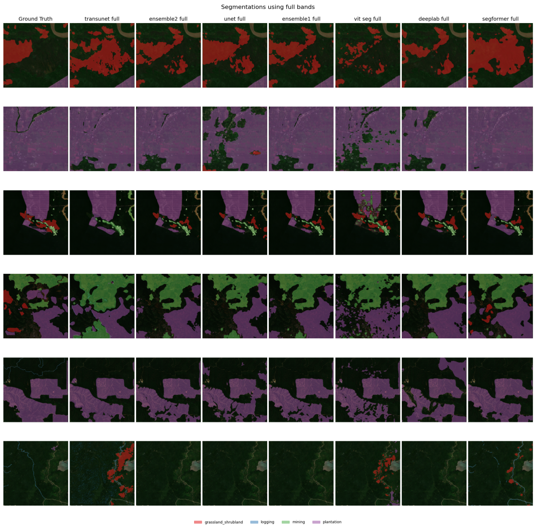

Figure 4. Segmentations using all channels

Training and validation performance

Overall performance

Most models converge to an F1-score of roughly 0.8 on training and 0.6 on validation, as seen in Figure 6 and Figure 7[3]. TransUNet's much lower performance is unexpected; it achieves only about 0.2 F1 on both sets.

Class-wise performance

Looking at the F1-score of each class in Figure 8 and Figure 9, we see that most models, except TransUNet, attain similar performance across classes. They perform slightly better on the training data, which is to be expected. All models seem to struggle with the classes `logging` and

[3] The figures only show the results from models trained on all channels, but results are similar for RGB as well. No post-processing is applied here.

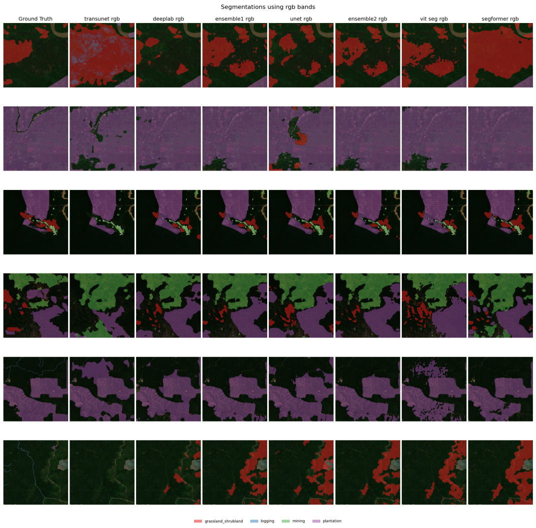

Figure 5. Segmentations using rgb channels

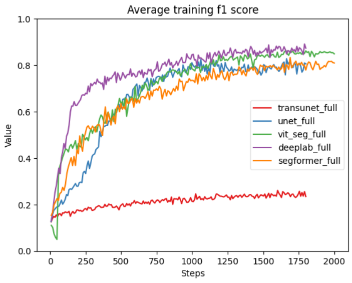

Figure 6. Overall training f1 score

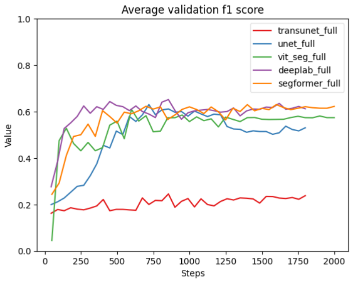

Figure 7. Overall validation f1 score

grassland_shrubland, more than plantation and mining.

The logging class consists of many thin lines (see the last two images in Figure 4), and models either ignore them entirely, or predict logging on similar, but unrelated features. For the grassland/shrubland class, the models tend to overpredict. Looking at the ground truth, it is hard to actually see what the grassland/shrubland area is, as it blends into surrounding vegetation, making it difficult to distinguish.

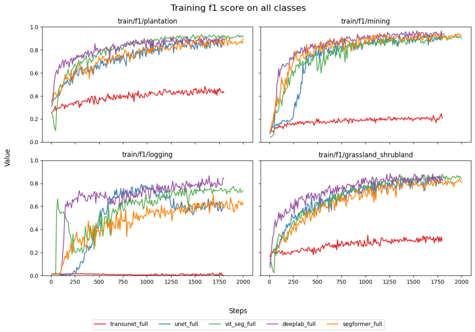

Figure 8. Training f1 score for all classes

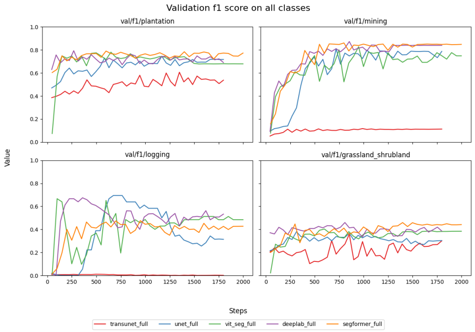

Figure 9. Validation f1 score for all classes

Training time

All the models we tried had varying sizes, and took different amount of time to train. As shown in Table 2 and Figure 10, the smallest models require only about one-third of the time training of the largest models TransUNet and ViT. Table 1 shows only a marginal increase in performance.

Model	RGB	Full
UNet	32.5	32.5
DeepLabV3+	26.7	26.7
Segformer	82.0	82.0
Vision Transformer	88.8	90.6
Transunet	105	105

Table 2. Number of parameters(millions) for models with RGB and all channels

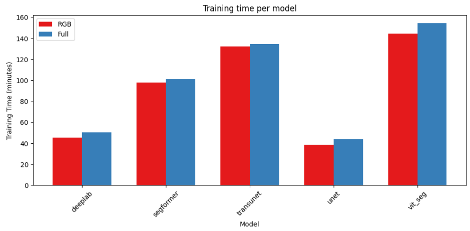

Figure 10. Training time for each model on 200 epochs

Competition performance

Our chosen model for the competition was DeepLabV3+ trained on all channels. It achieved an f1 score of **0.7367** on the validation data. On the public leaderboard it achieved a score of **0.5851**, and on the private leaderboard **0.5624**

Conclusion

DeepLabV3+ (full) achieved the best F1-score (**0.7367**), outperforming all other models, including the ensembles. The smaller models performed as well or better than the larger models, while using a lot less training time, making them more viable for further experimentation.

All models show weaker performance on logging and grassland_shrubland, suggesting possible class imbalance, or subtle patterns that are hard to detect. Because we did not check for class imbalance, we cannot definitively pinpoint the cause.

UNet and DeepLabV3+ served as reliable baselines, ViT proved the strongest pure-transformer segmenter. TransUNet, despite its huge capacity struggled on raw outputs but recovered much of its performance after minimum-area filtering.

Surprisingly our largest mode, TransUNet, underperformed the most in comparison to the other models. Huge models often "overfit" on limited data, since TransUNet need substantially more data to avoid overfitting, subsequently big networks are extremely sensitive to choices like learning-rate schedules and weight decay. Our relatively small, task-specific dataset, is insufficient to fully train such a heavyweight model. As a result, TransUNet "memorized" noise instead of learning generalizable patterns, leading to its very low raw F1.

TransUNet's poor result, despite its ImageNet-21K pretraining, can be caused by a few missteps. The encoder was frozen for 15 epochs (versus just 5 for ViT), so its rich pretrained features potentially never properly adapted before the decoder learned to segment. In hindsight, a more light unfreezing schedule, gentler learning-rate warmup, adjusted weight decay, larger effective batches, or extra regularization could help large models like TransUNet utilize their pretrained strengths. Unfortunately, time and computational constraints prevented us from exploring those avenues further.

References

Chen, Jieneng, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Yuille Alan L., and Yuyin Zhou. 2021. "TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation." *arXiv Preprint arXiv:2102.04306*.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al. 2021. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale." *ICLR*.

Iakubovskii, Pavel. 2019. "Segmentation Models Pytorch." *GitHub Repository*. https://github.com/qubvel/segmentation_models.pytorch; GitHub.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. "U-Net: Convolutional Networks for Biomedical Image Segmentation." https://arxiv.org/abs/1505.04597.

Xie, Enze, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. 2021. "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers." In *Neural Information Processing Systems (NeurIPS)*.