Deep Learning for Multispectral Image Segmentation: Methodologies and Results

Taravat Part

1 Introduction

Semantic Segmentation is a crucial computer vision task that involves the assignment of labels to every pixel in an input image. Currently, two prominent categories of deep learning models are being utilized to solve this problem: Convolutional Neural Networks and Transformers. While CNNs have traditionally been the go-to choice, the increase of data, specifically in the remote sensing domain, has pushed them to their limits. In contrast, transformer models have a larger capacity and great potential to increase model performance.

In this project, I carried out a series of experiments using a diverse range of CNN and Transformer-based models. Furthermore, I endeavored to enhance the performance of a specific Transformer model by incorporating an adversarial loss function, aimed at improving pixel classification at edges. This document presents the outcomes and insights from all these investigations.

2 Dataset

The provided dataset comprises multispectral images with 12 bands, each accompanied by its corresponding semantic segmentation mask. In total, this dataset contains 57128 samples. The input images are of size (12,64,64), while their associated semantic segmentation masks have dimensions of size (64,64).

I divided the dataset into three subsets: a training set, a validation set and a testing set, constituting 90%, 10% and 10% of the total dataset respectively.

Furthermore, minmax normalization is employed to standardize the pixel values within the input images. There are various upsides to this method:

- Normalization results in faster convergence of the neural network.
- Normalization can help prevent numerical instability during training. Larger feature values can lead to larger gradients and large weight updates. In other words, larger input values can lead to gradient explosion. Furthermore, if the algorithm perceives smaller values as less important, it may neglect their effect during optimization since larger feature values would dominate the optimization process.

Moreover, in our specific task, ensuring consistent scaling across different bands is crucial. This practice guarantees that pixel values in each channel are normalized to the same range. Consequently, it ensures that the neural network treats each pixel value and each band equally while preserving their relationships.

The following figure visualizes a sample in our dataset:

Figure 1 - Sample Input Image

Figure 2 - Sample Semantic Segmentation Mask

3 BACKGROUND

3.1 INTRODUCTION TO ENET – AN EFFICIENT NEURAL NETWORK:

ENet is a deep learning model specifically designed for semantic segmentation. Its primary goal is to deliver speedy inference while keeping computational costs at a minimum. This architecture comprises two main components: an encoder and a decoder. The encoder's role is to extract features from the input image by applying convolutional and max pooling layers. Meanwhile, the decoder takes these features, upsamples the feature maps, and produces a segmentation mask as output that matches the original input image's dimensions.

Downsampling plays a crucial role in convolutional neural networks. This is because the kernels in a convolution layer have a limited local receptive field and cannot capture global information from the entire image. However, by progressively downsampling the feature maps through convolutional layers in a deep neural network, deeper layers can achieve a larger receptive field, even when using kernels of the same size as those in the shallower layers.

Nonetheless, excessive downsampling comes with a major drawback, which is loss of spatial information like exact edge shape. This limitation becomes particularly problematic in tasks like semantic segmentation, where the task is to assign a class to each pixel in the input image and create a mask image of the same size as the input. Furthermore, excessive downsampling often necessitates a higher number of layers, leading to a substantial increase in model parameters, and subsequently increasing the computational costs.

The tackle these challenges, ENet offers a number of solutions to achieve both high accuracy and low computational costs:

- **Feature map resolution:** The indices of elements chosen in max pooling layers are saved and subsequently utilized to produce sparse upsampled maps in the decoder. Additionally, excessive downsampling is mitigated as much as possible.

- **Early downsampling:** The first two blocks of ENet heavily reduce the input size. The idea is that visual information is highly spatially redundant, and thus can be compressed into a more efficient representation.
- **Factorizing Filers:** ENet employs factorization of convolution filters. The authors propose breaking down each NxN convolution into two smaller ones following each other: one with a 1xN filter and the other with an Nx1 filter. This factorization results in significant speed improvements and reduces the number of model parameters.

- **Dilated Convolutions:** To expand the receptive field of kernels without increasing the number of parameters, ENet uses dilated convolutions. This technique enhances the model's ability to capture larger contextual information without adding computational complexity.

Enet Architecture consists of various of blocks which are summarized below:

- **Initial Block:** ENet begins with an initial block that combines standard convolution and max-pooling layers. This block efficiently reduces the spatial dimensions of the input while capturing essential features.
- **Downsampling Block:** Downsample layers are used to extract features while reducing the spatial dimensions gradually. These blocks utilize either regular, assymetric or dilated convolution filters.
- **Upsampling Blocks:** ENet's architecture includes upsampling blocks that restore the spatial dimensions of the feature maps.
- **Regular blocks:** These blocks neither upsample nor downsample the feature map. They just create several representations of the input feature map.

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
$4\times$ bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
Repeat section 2, without bottleneck2.0		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

Figure 3 – Efficient Neural Network (ENe) Architecture

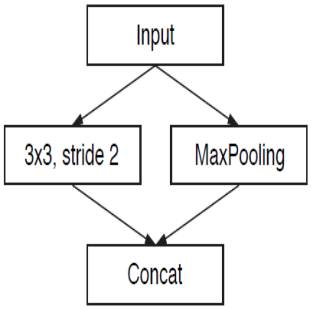

Figure 4 - Upsampling block

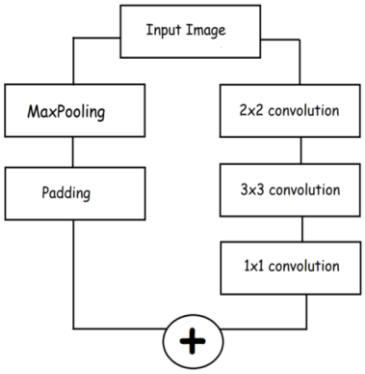

Figure 5 - Downsampling Block

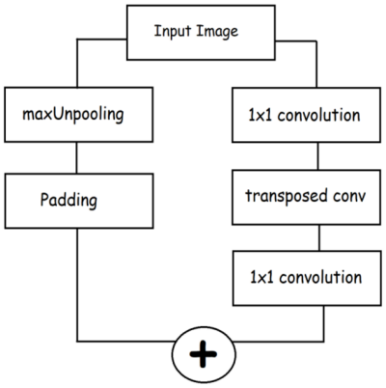

Figure 6 - Initial Block

3.2 UNET:

UNet is a convolutional neural network designed for biomedical image segmentation. Its U-shaped architecture consists of an encoder pathway to capture context and extract features, and a decoder to achieve high-resolution segmentation masks. The skip connections between encoder and decoder paths enable precise localization of objects by preserving fine-grained spatial information.

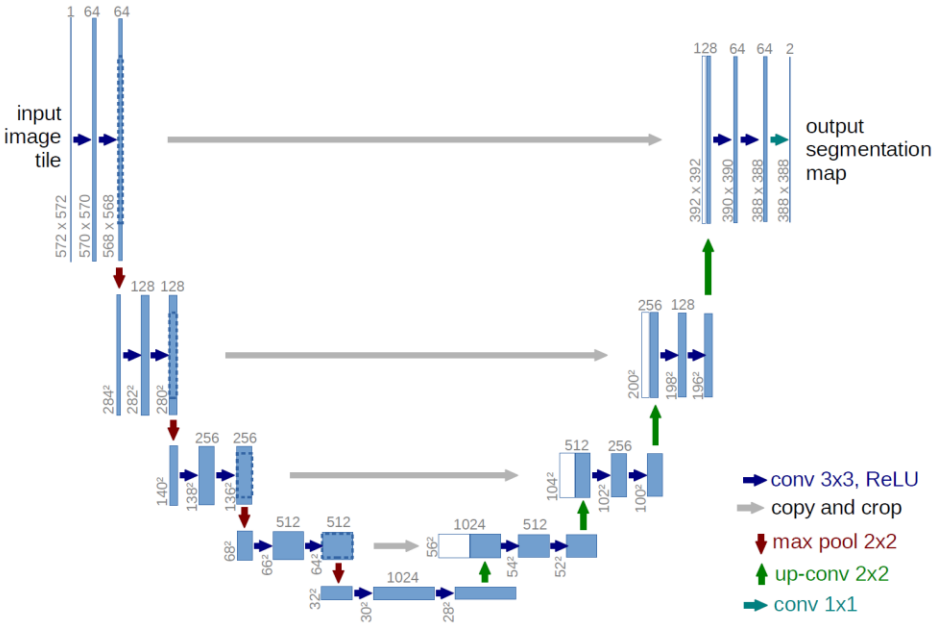

Figure 7- UNet Architecture

3.3 TRANSUNET:

TransUNet is a deep neural network model originally created for medical image segmentation. While CNN models, such as UNet, have achieved significant success in a variety of image segmentation tasks, they have limitations when it comes to modeling long-range dependencies due to the localized nature of convolution filters. In contrast, transformers were initially

introduced for sequence-to-sequence prediction and leverage self-attention layers, which enable the capture of global context in an image.

TransUNet merits the advantages of both transformers, which excel at capturing global context, and convolution layers, renowned for their ability to capture spatial patterns within images. The models architecture consists of two main components: a hybrid encoder and a decoder.

The model's architecture starts with convolution layers, which generate a low-resolution feature map from the input image. Then, 1x1 patches are extracted from the final feature map and patch embedding is applied on them. The extracted feature embeddings go through a series of transformer blocks. The self-attention layers within transformer blocks facilitate modeling the global interactions among all pixels in the feature map.

Following this, the representations generated by the hybrid encoder proceed through the decoder, which resembles the decoder of UNet. The decoder mainly consists of upsampling layers responsible for generating the output mask.

Furthermore, intermediate feature maps generated by the convolution section of the encoder are integrated with the intermediate feature maps generated by the decoder. This empowers the model to preserve fine-grained spatial information, which is particularly important for semantic segmentation tasks.

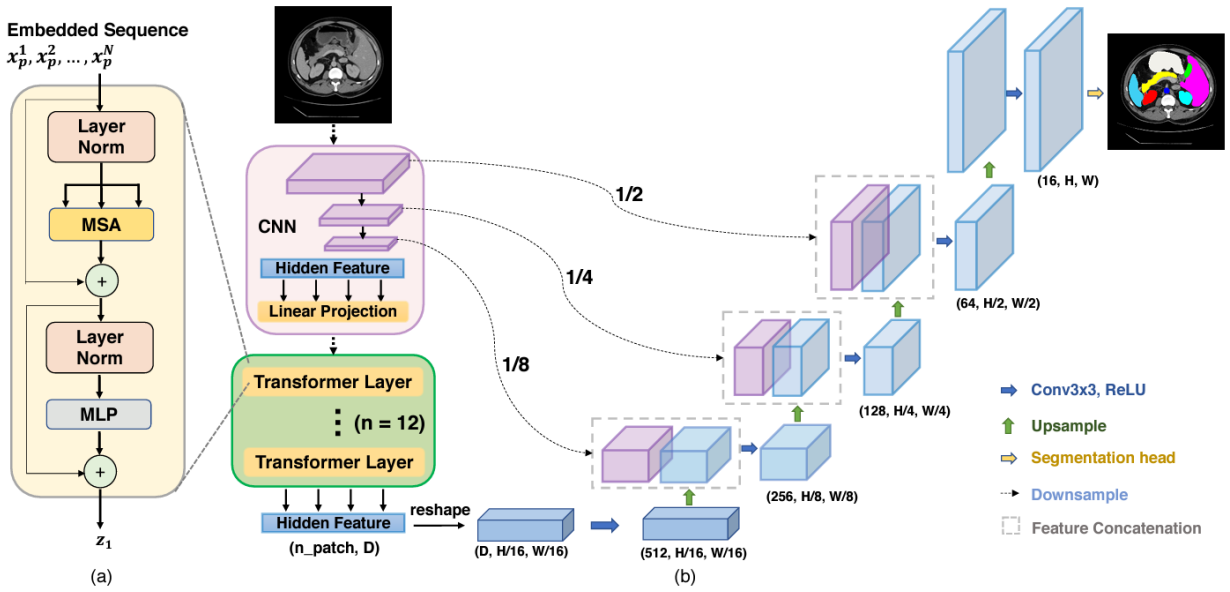

Figure 8 - TransUNet Architecture

4 EXPERIMENTS

4.1 TRAINING ON MULTI-SPECTRAL IMAGES:

I conducted experiments with four distinct types of models in this project. Initially, I trained two CNN-based models named ENet and UNet, utilizing cross-entropy loss and the Adam optimizer. Subsequently, I proceeded to train a hybrid CNN-Transformer model known as Transunet. My experimentation with this model took two forms: training it from scratch and leveraging pretrained weights, fine-tuning it on our specific dataset.

To further enhance the quality of the generated semantic segmentation masks, I employed TransUNet as a generator and integrated a discriminator into the process, building a generative adversarial neural network. The discriminator's role was to receive the edge maps of the target segmentation masks (real data) and the edge maps of the generated segmentation masks (fake data), and classify these edge maps as either fake or real. Canny edge detector was use to extract the edge maps. This approach involved training both the generator and discriminator with an adversarial loss in addition to the initial cross-entropy loss. The intention behind this strategy was to guide the model in producing segmentation masks with more realistic edges and improve the evaluation metrics.

The following table demonstrates the performance of the employed deep models:

	Accuracy	F1 score	Kappa	MIOU
ENet	78.6	79.08	0.63	0.61
UNet	88.9	88.9	80.3	75.8
TransUNet	88.9	89	80.1	76
TransUNET-Finetuning	89.15	89.31	80.83	76.2
Adversarial TransUNet	87	87.3	77.2	73
TransUnet – utilizing 3 channels				

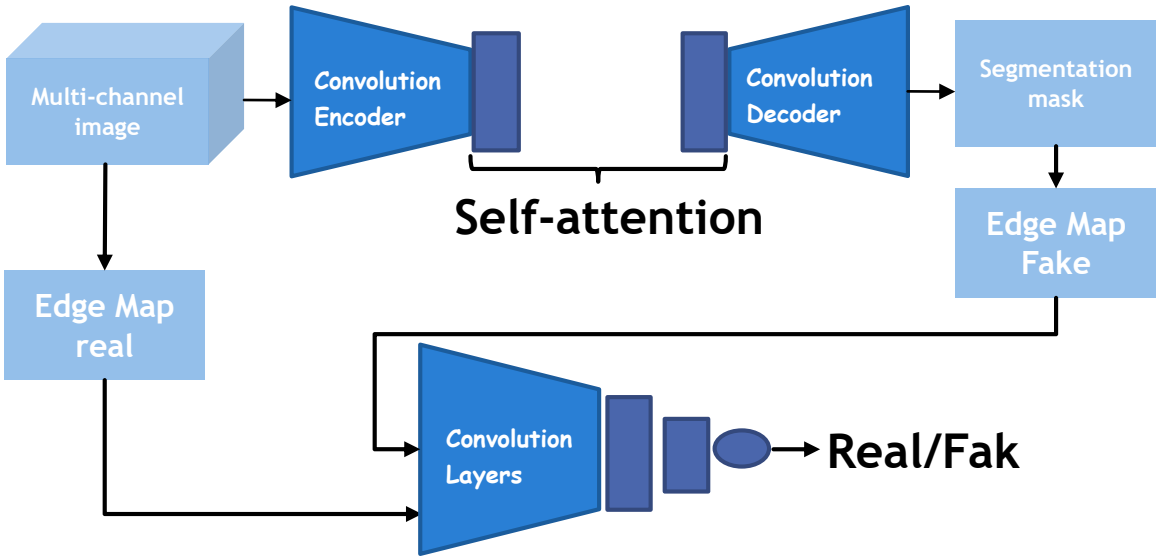

Figure 9 - Generative Adversarial Transformer

The following images demonstrate the training process of all these models over ten epochs:

UNet training progress:

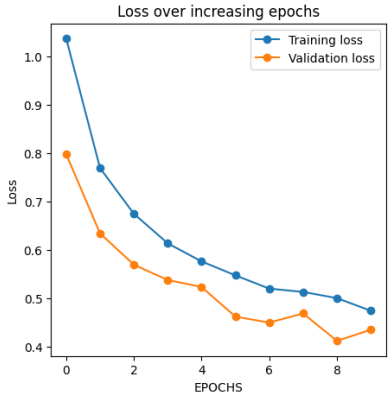

Figure 10 - Unet - Cross Entropy Loss per Epoch

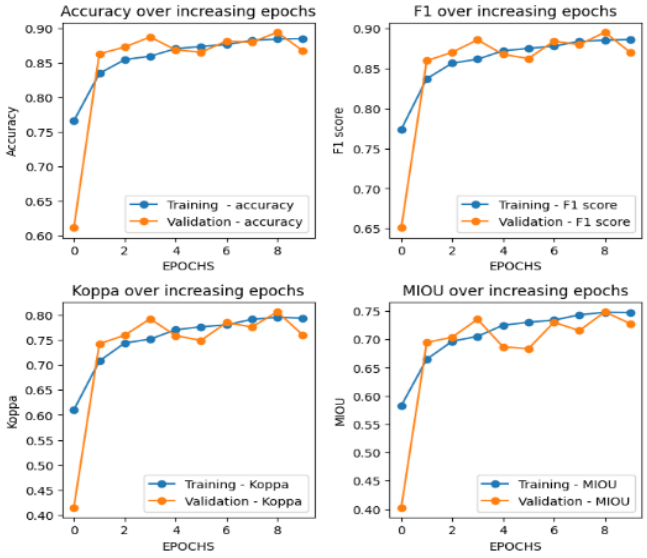

Figure 11 - UNet - Evaluation Metrics per Epoch

ENet training progress:

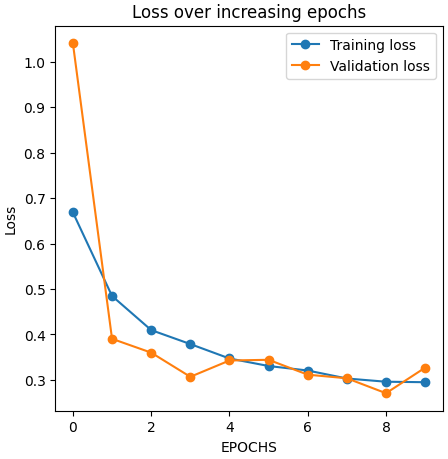

Figure 12 - ENet - Cross entropy Loss per Epoch

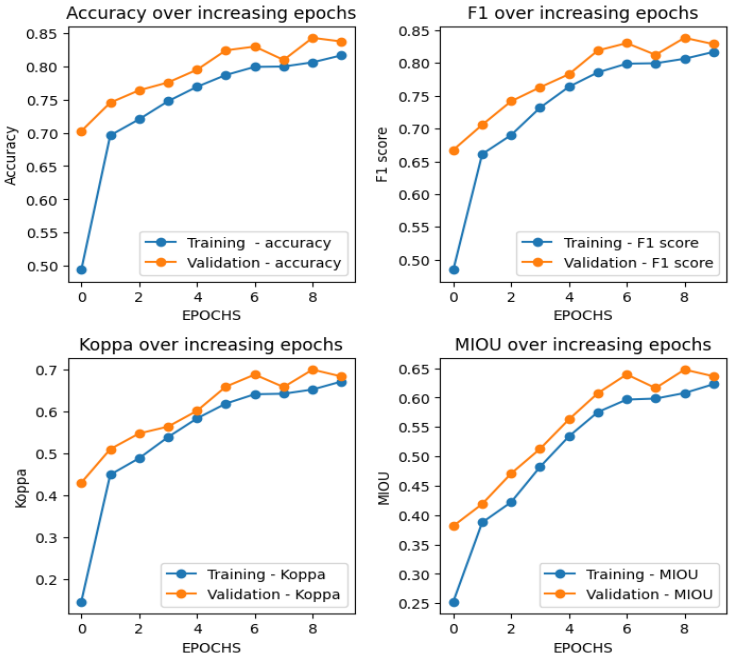

Figure 13 - ENet - Evaluation metrics per Epoch

TransUNet training progress:

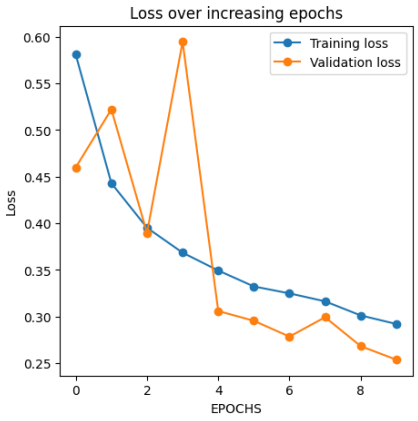

Figure 14-TransUNet - Cross entropy Loss per Epoch

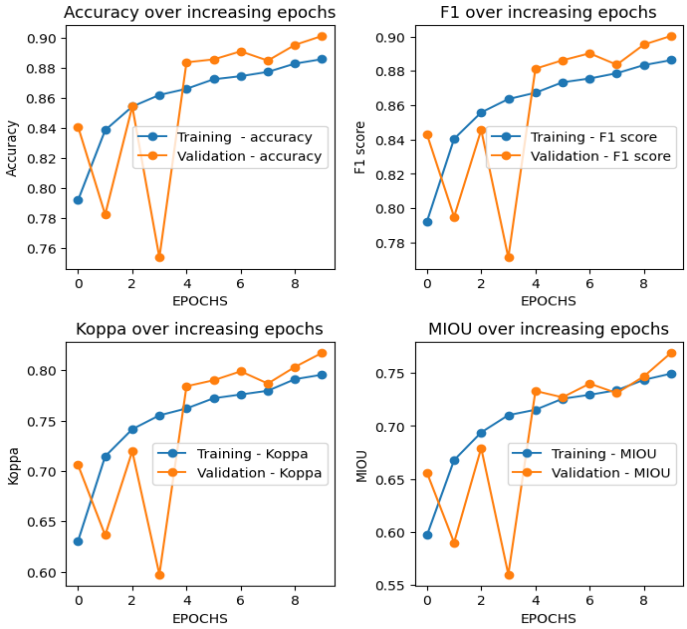

Figure 15 - TransUNet - Evaluation Metrics per Epoch

TransUNet Fine-Tuning progress:

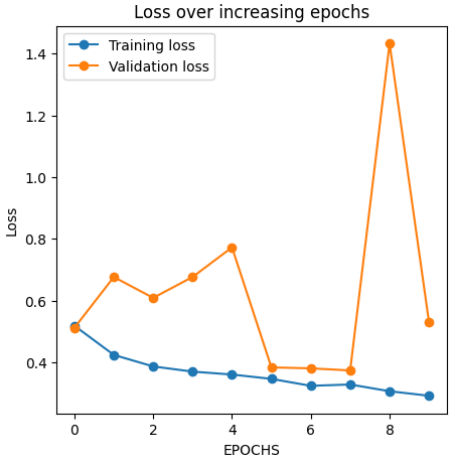

Figure 16 - Finetuning TRansUNet - Loss Value per Epoch

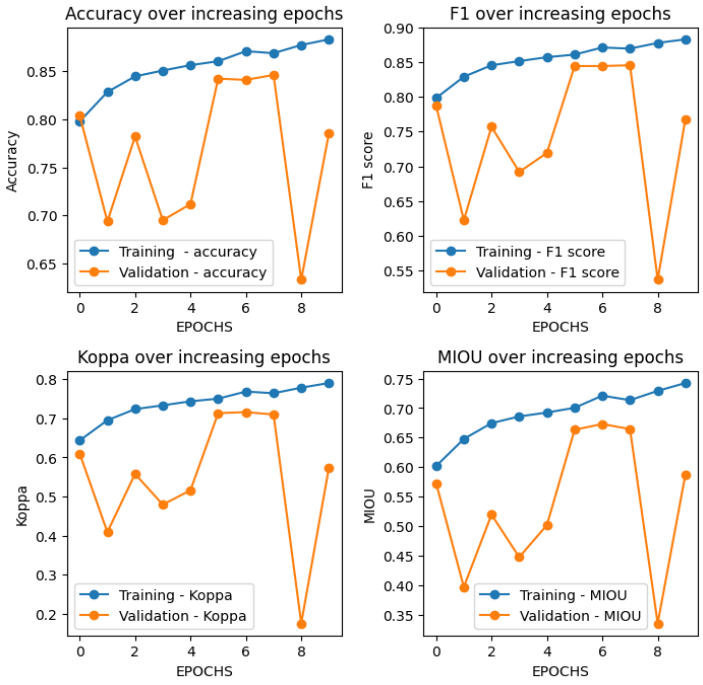

Figure 17 - Finetuning TRansUNet - Evaluation metrics per Epoch

Adversarial TransUNet training progress:

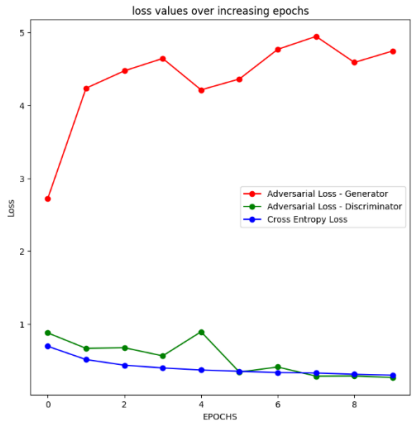

Figure 18 - Adversarial TransUNet - Loss per epoch

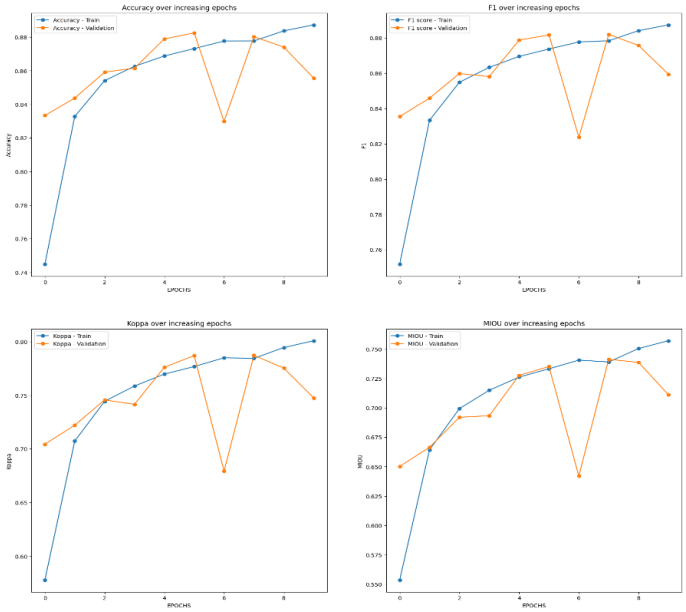

Figure 19 – Adversarial TransUNet – Evaluation metrics per epoch

4.2 Training on RGB Channels:

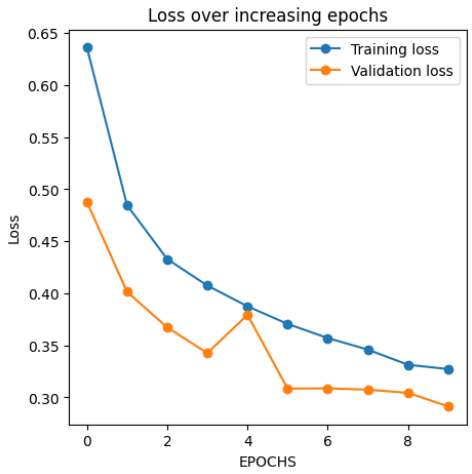

Figure 20 - TransUNET on RGB channels - Loss value per epoch

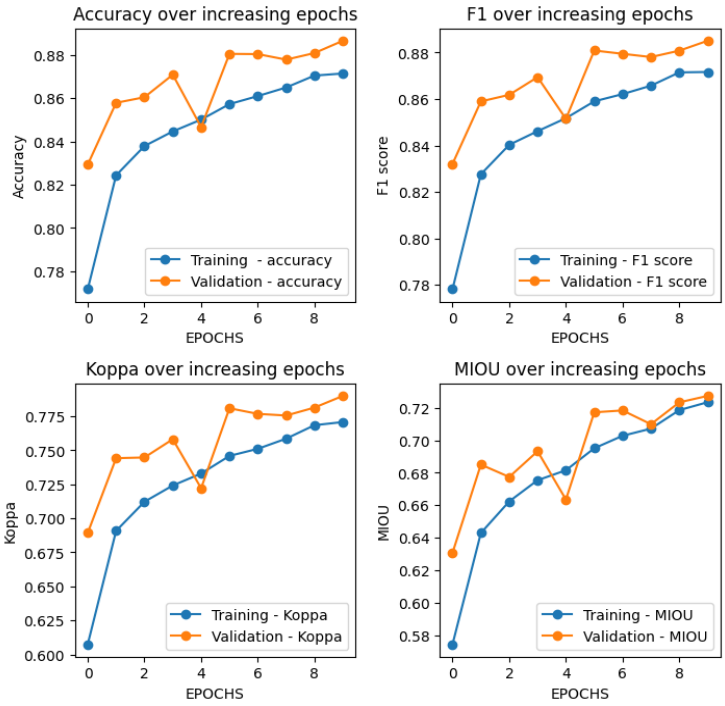

Figure 21 - TRansUNet on RGB channels - Evaluation Metrics per Epoch

5 ANALYSIS

- The training process of UNet demonstrates that the loss value was mostly decreasing over epochs, and evaluation metrics were mostly increasing over epochs. However, the figures clearly indicate that the model was reaching its limit after ten epochs. The flattening shape of the figures demonstrates that the UNet model converges after ten epochs and does not improve significantly by increasing the number of epochs.
- The training process of ENet demonstrates that ENet achieved a lower loss value on the training data compared to UNet after ten epochs. However, ENet performs worse than UNet on the test data, indicating overfitting.
- Due to the large capacity of transformers for learning complex patterns, I expected them to perform better than convolutional models, which proved to be true. The loss value figure on the training data shows a significant drop in the initial epoch. Furthermore, all the evaluation figures for the training data followed an upward trend. None of the figures displayed a flattening trend, affirming that continued training for more epochs would undoubtedly yield better results. However, the validation figures demonstrated fluctuations, indicating that transformers are prone to overfitting when not trained with a sufficient number of samples.
- Fine-tuning TransUNet reveals that the model is overfitting to the training data. Despite the evaluation metrics showing improvement over successive epochs on the training data, the metrics for validation data exhibit significant fluctuations. Specifically, the accuracy on validation data starts at 80% in the initial epoch and remains at 80% by the 10th epoch. This behavior strongly suggests that the model is overfitting. Although the model attains reasonable metric values, these results are primarily attributable to the prior knowledge of the pre-trained model. Training it on our dataset is causing overfitting.
- According to the training of the adversarial TransUNet, all the metrics follow an upward trend with a steeper slope in comparison to the previous models. Therefore, I am confident that the adversarial TransUNet would outperform the previous models if it were trained for a larger number of epochs. However, the training time for this model is longer than that for the other models, as it requires the training of two subnetworks and the optimization of a larger number of parameters.
- In the last experiment, I utilized only three channels for training TransUNet. The evaluations on test data illustrate that the model's performance more or less remained the same when compared to the case where we used all the channels. The reason behind this may be twofold:
    - Not all the channels may contain informative features to improve model performance. Therefore, we may discard the irrelevant channels.
    - Our model architecture may not be efficient enough to fully utilize the information from all the channels. We should consider implementing some new techniques to make the most of all the channels to our advantage.

6 FUTURE WORK

Many other ideas can be tried out to check if the model's performance on this particular dataset can be enhanced. Here are a few methods that I think we can explore in the future:

- **Adding a patch-level classification loss function:** Currently, the self-attention layers in TransUnet generate feature vectors, each representing one of the image patches. We can take the feature vectors that are outputted by the last transformer module and perform classification on top of them. In this way, the model will not only perform pixel-level classification but also patch-level classification. Therefore, the model will learn how to update its parameters to generate not only a correct segmentation mask at the output but also feature vectors that are representative of the correct class. We can simply determine the ground-truth class of each patch by performing a max or mean operation on the labels of each patch.
- **Introducing an Edge-Aware Loss Function:** To enhance model performance, we can compute the mean squared error on the real edge maps and the generated edge maps. This approach will encourage the model to produce masks with accurate edges.
- **Refining self-attention mechanism:** there are three main ideas we can consider:
    - **Spatial Attention Modules:** Firstly, we can implement modules that compute attention scores spatially. These modules will generate representations for each patch, considering how much attention each patch should give to other patches within the same channel.
    - **Inter-Channel Attention Modules:** Additionally, we can introduce attention modules designed to calculate attention across channels. These modules will determine the degree to which each channel should focus on other channels. In this module, each channel would constitute one of the input tokens.
    - **Inter-channel Positional Transformer Module:** To further enhance our model, we can incorporate a transformer module that receives tokens from different channels located at the same position. This module will exclusively calculate attention among these tokens, ensuring a more nuanced understanding of spatial relationships.

- Finetuning other pretrained transfoerms like SwinUNetR.