Assignment 2 - Deep Learning for Visual Computing

Vladimir Panin, Philipp Rettig, Group 50

May 27, 2024

1 Questions

1.1 What other architectures for segmentation apart from the SegFormer have you learned in the lecture and how do they differ from the SegFormer?

Apart from SegFormer, the lecture covered other segmentation architectures such as Fully Connected Networks , U-Nets, and Feature Pyramid Networks. While SegFormer adopts a transformer-based architecture, FCN, U-Net, and Feature Pyramid Network use different approaches. Fully Connected Networks makes use of a fully convolutional network to perform pixel-wise classification. U-Net utilizes a U-shaped architecture with skip connections between encoding and decoding stages to capture both local and global features effectively. Finally, a Feature Pyramid Network utilizes a pyramid structure where most of the computations are performed in the encoder with a fairly lightweight decoder. These architectures rely on convolutional operations and don't make use of attention

1.2 Why are those architectures using down- and up-sampling instead of keeping the same resolution?

The idea behind down- and up-sampling is to learn high-level as well as also low level features. Down-sampling uses pooling and strided convolution to make use of the global context of the picture. Up-sampling uses L-interpolation and transposed convolution to let the network also learn local feature. By the integration of local details and global patterns the network ensures consistent image segmentation despite different object sizes and positions.

1.3 What is the purpose of pre-training and fine-tuning? For which use-cases is it especially beneficial?

Pre-training deals with learning general patters from a comparably big dataset. After pre-training fine-tuning can be performed on a task-specific dataset, which is generally smaller than the general dataset. The weights of the big dataset are reused for the downstream task. Hence, as a result the weights are only slightly modified compared to the pre-trained one. Usually, a smaller learning rate and few epochs are used. This way computational resources can't be saved and previous models reused to adjust them to the current task.

1.4 What have you observed in your experiments above, how has using pre-trained weights influenced the performance and why? Do you see any differences between the two fine-tuning option of part 6?

Setup:

- Batch size: 16 for the cityscapes data set, 64 for the oxford dataset
- Optimizer: Adam with AMSGrad
- Learning rate: 0.001, learning rates of 0.0001 and 0.0005 were also experimented with in the fine-tuning phase
- Loss function: Cross-Entropy loss
- Number of epochs: 28 for the oxford data set, 40 for the pretraining on the cityscapes data set

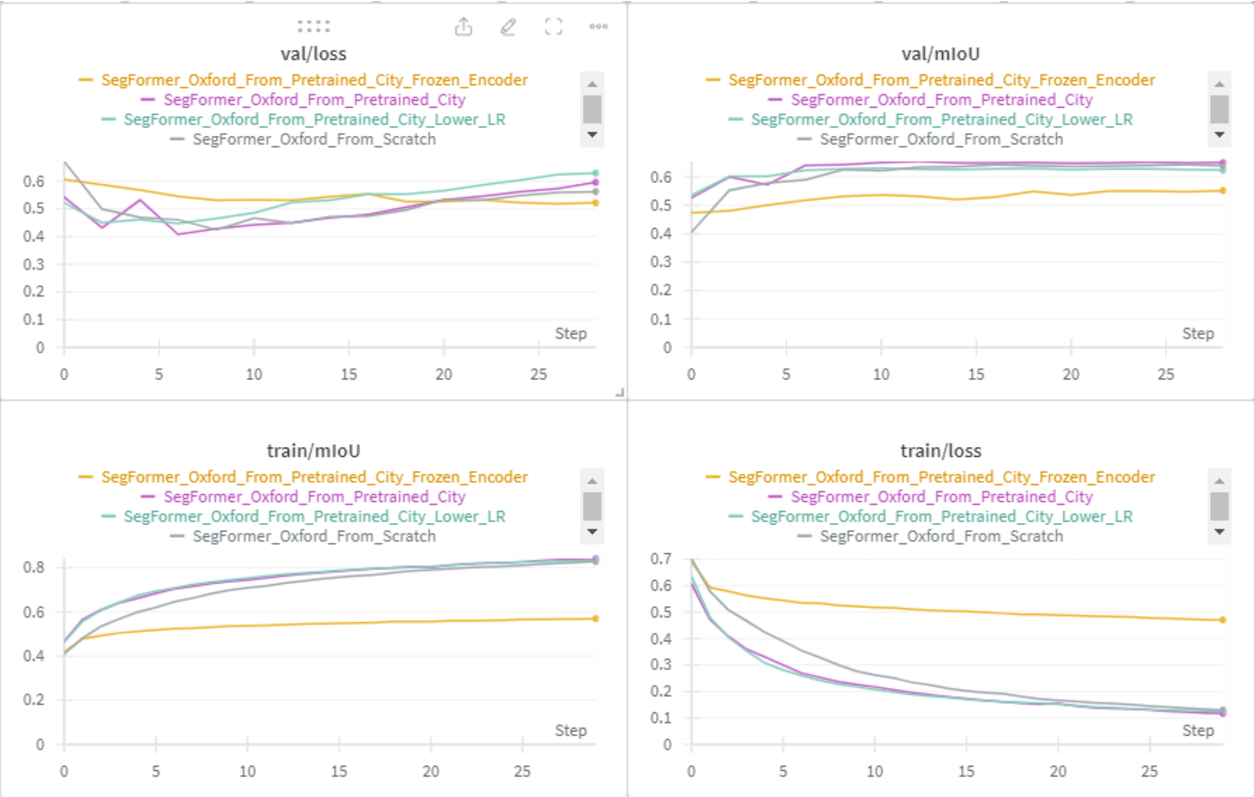


Figure 1: Segformer - comparison of training strategies

The results of our experiments are displayed in figure 1. Notably, the best performance, as measured by the mean intersection over union, was achieved using fine-tuning strategy a. Surprisingly, training the oxford data from scratch yielded results that were nearly comparable. Typically, one would expect that fine-tuning a pretrained model would give a significant performance boost on the validation data, as the models ability to generalize should improve. Possibly the model is reaching its limits on the oxford data set or there are other factors like insufficient regularization in the pretraining phase or choosing the appropriate parameters for optimization.

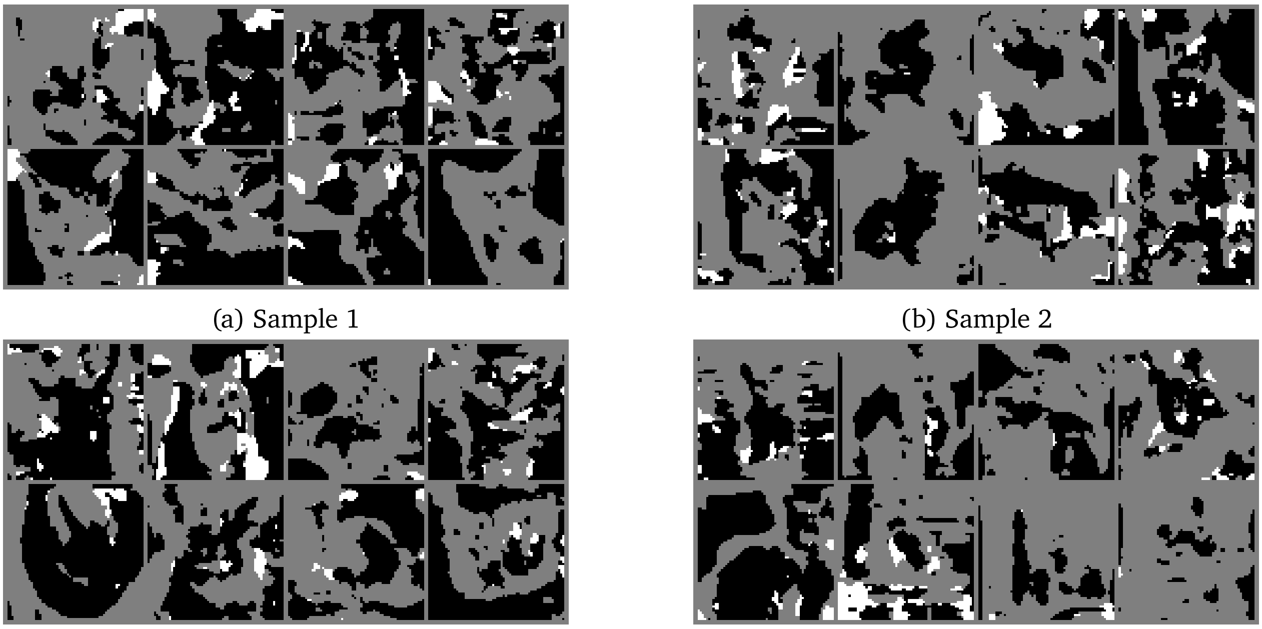
Table 1: mIoU

Training Strategy	mIoU (%)
From scratch	63.88
Pretrained weights	64.99
Pretrained weights with lower lr	62.38
Pretrained weights with frozen decoder	55.17

The strategy of freezing the decoder weights led to a noticeable drop in performance. This might not be totally unexpected, as the decoder in the segformer model is a crucial part in upscaling the learned features back to the original dimensions.

Furthermore, choosing a learning rate of 0.001 seemed to give the best results, even for fine-tuning. Attempts to reduce the learning rate led to improved performance on the training set, but failed to generalize to the validation set.

Images of predicted masks of some validation samples are illustrated in figure 2.



(a) Sample 1

(b) Sample 2

(c) Sample 3

(d) Sample 4

Figure 2: Predicted masks of validation samples