Transfer learning for Domain Adaptation

John Huang, Alex Kumar, Junyao Zheng

Abstract

In this study, we leverage the MMSegmentation library to explore various backbone models on the diabetic macular edema (DME) dataset. Our approach includes finetuning candidate backbone models to achieve optimal performance on the DME task. Given the challenges of semantic segmentation, especially in medical imaging with limited datasets, we employ transfer learning with fully convolutional network (FCN) model architectures. Our contributions extend from model training to modifications in the experimental pipeline. We explore enhancements in data preprocessing, loss functions, and model architecture to optimize performance. The study conducted on the DME dataset demonstrates the adaptability and effectiveness of transfer learning for semantic segmentation tasks.

1 Introduction & Motivation

We are trying to perform semantic segmentation, or pixel-wise image classification, using the *MMSegmentation* library [1] which contains pre-trained segmentation models. We will test out some of the various backbone models on the diabetic macular edema (DME) dataset. After exploring the dataset and the models, we will finetune candidate backbone models to try and achieve high performance on the task of identifying diabetic macular edema (DME) in ocular coherence tomography (OCT) b-scan images. After the finetuning step, our team will explore possible improvements in the experimental pipeline, such as modifying the data preprocessing pipeline, loss function, or model architecture.

Semantic segmentation is done by fully convolutional network (FCN) model architectures. Unlike simpler image classification tasks, good performance is hard to achieve without transfer learning. Transfer learning is the process of copying weights from a model trained on general images to initialize an application-specific model allowing for faster convergence. This is needed in fields such as medical imaging since the datasets are usually very small. There are many different architecture variations on convolutional networks, like U-Net, Mask R-CNN, Semantic FPN, and more. There are also the usual data transformations, such as horizontal flipping and adversarial patches, to augment the dataset and combat overfitting.

2 Related Works

We started by conducting a brief literature review on the field of image segmentation applied to medical imaging. Research has already shown that a modified U-Net model can work well on the segmentation of retinal fluid in OCT b-scans when trained from scratch on densely sampled volumetric OCT c-scans [12]. Similarly, research using MMSegmentation has proven the ability of the library on adapting pre-trained weights to medical imaging [13]. Other research focuses on analyzing how the source domain impacts the outcome on the target domain [11]. As these works focus on performing instance segmentation with a single machine learning model, we were more interested in exploring the change in performance of different architectures and methods to obtain optimal performance when adapting them to new domains.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

3 Methodology: MMSegmentation Pipeline

OpenMMLab is an open-source computer vision ecosystem based on PyTorch that includes MMSegmentation [1], a library specifically built for semantic segmentation that establishes a framework for modifying and configuring pre-made and custom semantic segmentation CV models. This framework revolves around the concept of configuration files, which are simply dictionaries with specific keywords that are eventually parsed into underlying PyTorch models. These configuration files also include specifications on how the dataloading, training, and validation processes should be run. The MMSegmentation framework breaks down semantic segmentation models into interchangeable components. Among the various components, the "Backbone" and "Decode head" are the most important ones for our use case. The Backbone, traditionally called the encoder, transforms the inputted images into feature maps. The Decode head transforms the feature maps into the final segmentation masks.

The main reason for using the MMSegmentation library is its selection of pre-trained semantic segmentation models across 11 different backbones, numerous different Decode heads, and 23 semantic segmentation datasets. The process of utilizing these pre-trained models is the following: downloading the pre-trained weights, sourcing the configuration file for the specific pre-trained model, modifying the configuration to fit our new dataset, and training the new model.

After parsing the data into a format usable by MMSegmentation, a Python script was written to modify the configuration file then build, train, and save the resulting model. The build, train, and save steps were accomplished using the Runner class from MMEngine (a foundational library of OpenMMLab), which takes in the configuration file and runs the train and validation steps on the generated model as specified. The main modifications to the configuration file consist of getting the model to load the target dataset correctly. Further modifications such as adding to the training pipeline, modifying the Decode head, and modifying the training data loader allow for different data augmentations, loss functions, and batch sizes respectively.

3.1 Dataset Used for Pretraining

The exploration scope in this paper is limited to models trained on medical imaging datasets with the assumption that transferability is greater between similar domains. Of the medical imaging datasets available through MMSegmentation, these four were the only ones with pre-trained models:

- The Digital Retinal Images for Vessel Extraction (DRIVE) [8] dataset contains RGB retinal images with annotations of blood vessels.
- The Structured Analysis of the Retina (STARE) [10] dataset also contains RGB retinal images with annotations of retinal diagnoses, blood vessel segmentation, artery or vein labeling, and optic nerve location.
- The Child Heart and Health Study in England (CHASE) dataset consists of retinal images with blood vessel annotations.
- The High-Resolution Fundus (HRF) [9] dataset consists of retinal fundus images with annotations on the optic disc, and blood vessels.

3.2 Decoder Head

Among all the backbones in MMSegmentation, U-Net is the only backbone with models that are pre-trained on the medical imaging datasets mentioned in section 3.1. For each of these datasets , the U-Net Backbone comes with models pre-trained on three different decoder heads:

- Fully Convolutional Networks (FCN) [5]: Consists of all convolutional layers, allowing the model to perform pixel-wise prediction on images.
- Pyramid Scene Parsing Network (PSPNet) [3]: Utilizes pyramid pooling module on top of a deep convolutional neural network, enabling the capture of context at multiple scales.
- DeepLabV3+ [4]: Employs atrous (dilated) convolution layers to increase the area covered by filters, while maintaining a lower computation cost.

This combination of three decoder heads and four datasets provides a total of 12 different pre-trained models to be tested.

3.3 Tuning Parameters

3.3.1 Batch Size

Varying the batch size during training creates a trade-off between computation speed and convergence stability. With a larger batch size, parallel processing of GPUs can effectively speed up the training process, but the model would also risk over-fitting or being unable to escape local minima. With a smaller batch size, the model risks not sufficiently learning and being unable to converge on an appropriate solution. The goal was to explore different batch sizes in order to find the balance between computation speed and convergence behavior.

3.3.2 Data Augmentation

Models were trained with and without data augmentation. Our data augmentation pipeline included random resizing of scale ranging from 0.5 to 2.0, random cropping to a 256 x 256 image, and a random horizontal flip with 0.5 probability. A 256 x 256 crop was used to ensure that the resulting image had enough surrounding context for the fluid regions to be identified. Only horizontal flipping was used since OCT images capture the retina layers going horizontal across the image when taken normally, thus vertical flipping would introduce unnecessary complexity into the data for the model to learn. Without data augmentation, the data was still resized by a scale of 0.5. This was due to hardware capabilities as training with the full images either did not start on some machines or had estimated completion times of over an hour as compared to a few minutes when scaled down.

3.3.3 Loss Function

In the DME dataset, there is a significant class imbalance between the fluid class and the background, so we suspected that cross entropy loss would underperform compared to Dice loss. Therefore, we performed experiments with cross entropy loss, Dice loss, and a hybrid loss function where the weights of Dice loss to cross entropy loss is 3 to 1.

3.3.4 Other Training Parameters

Across all models, an SGD optimizer was used with a learning rate of 0.01 and with no appreciable decay. Models were trained for 200 epochs unless otherwise specified.

3.4 Evaluation Metrics

Two of the most common evaluation metrics were employed to measure the performance of the models, Intersection over Union (IoU), which is defined by $\frac{\text{Area of Intersection}}{\text{Ground Truth Area + Predicted Area - Area of Intersection}}$ and Dice Score, which is defined by $\frac{2\times\text{Area of Intersection}}{\text{Ground Truth Area + Predicted Area}}$.

4 Target Dataset for Transfer Learning

The target DME dataset consists of OCT b-scan images of eyes that are either healthy or inflicted with DME [6]. DME is a condition where fluid accumulates within the retinal layers causing their thickening. OCT b-scan images provide a high-quality cross-section which is crucial for identifying the locations of fluid-filled regions caused by DME. The DME dataset contains 110 OCT images of size 768 x 496 with two sets of segmentation annotation masks for the fluid-filled regions from two different experts (though only one is used as the ground truth for training and testing). An 80/20 split is used to split the total images between the training and testing datasets. This allows both datasets to contain images of healthy eyes and of fluid regions with varying sizes and shapes.

5 Experimental Results

5.1 Training Without Pre-trained Weights

As an initial investigation, a U-Net+FCN model was trained on the DME dataset without any pre-trained weights as a starting point, instead, Kaiming initialization was used. In training this model data augmentation and cross-entropy loss were used. It took about 1000 epochs for the model to

produce non-zero fluid scores, and a total of 2000 epochs were used for training. As seen in the results shown in Table 1, the resulting model is comparable to models trained with transfer learning but the time it takes to reach this level of performance is a magnitude larger.

Table 1: Performance of U-Net+FCN with and without pre-trained weights

Pre-train Dataset	IoU	Dice	Accuracy
None	31.00	37.80	47.33
STARE	36.72	54.3	66.28

5.2 Model Tuning

5.2.1 Effect of Batch Size on Performance

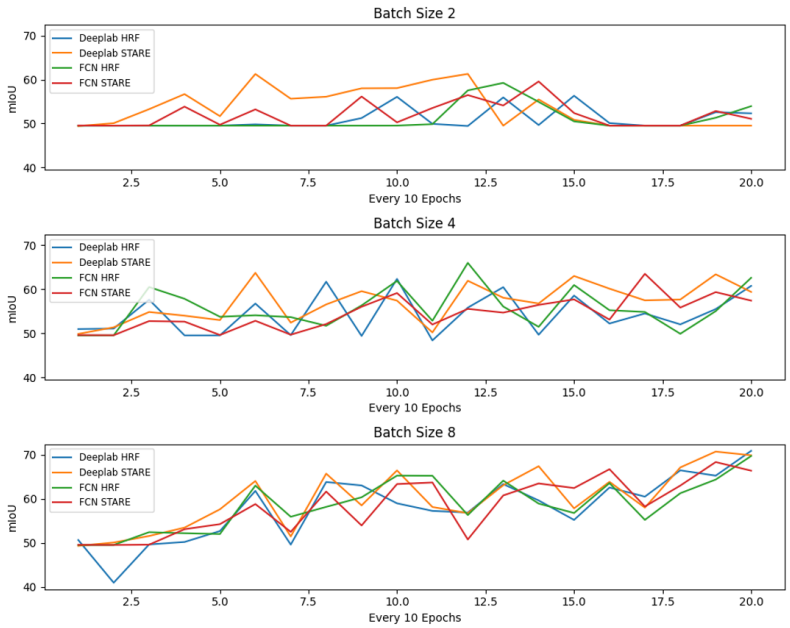

Figure 1: mIoU of models trained with different batch size

Table 2: Performance of U-Net+DeepLabV3+ models with different batch size on fluid class

Batch Size	IoU	Dice	Accuracy
2	22.56	38.20	56.74
4	27.44	43.72	71.88
8	41.48	59.04	71.04

Our investigation into the impact of batch size on model performance revealed a linear relationship. However, this trend ceased to improve beyond a batch size of 8 due to hardware constraints. Possible explanations include: noise reduction, where larger batches likely contributed to noise reduction in gradient estimates, fostering smoother convergence during training, and greater generalizability, where the linear improvement suggests that larger batches enhanced the model's generalizability, providing a more accurate estimation of the gradient.

5.2.2 Impact of Data Augmentation

Our experiments with data augmentation demonstrated a notable increase in performance. We explain that it helped address overfitting. Given the limited size of the DME dataset, Table 3 suggests that

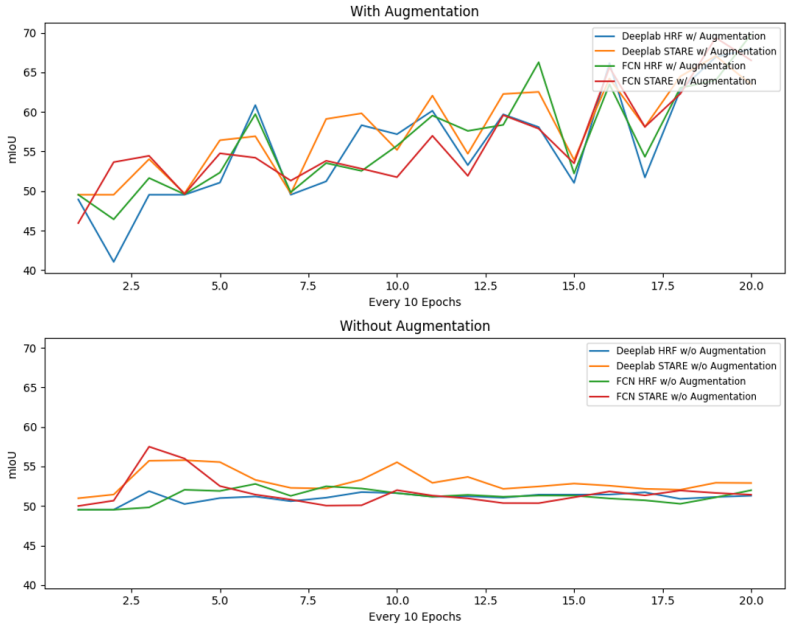

Figure 2: mIoU of models trained with and without data augmentation on fluid class

Table 3: Performance of U-Net+DeepLabV3+ CHASE models with and without data augmentation

With Augmentation	IoU	Dice	Accuracy
Yes	41.48	59.04	71.04
No	12.12	23.12	23.42

data augmentation played a crucial role in combating overfitting by generating more diverse training samples. Through observation on Figure 2, we can also note that data augmentation essentially allows the model to begin converging, while mIoU of models trained without data augmentation does not show a clear trend of convergence.

5.2.3 Influence of Loss Function

Table 4: Performance of U-Net+PSPNet models with different loss function on fluid class

Loss Function	IoU	Dice	Accuracy
CE	45.29	65.25	62.34
Dice	42.66	61.04	59.81
Hybrid	41.65	66.38	58.81

As shown in table 4, we were not able to spot a significant difference in performance when training with different loss function. This could possibly attribute to the fact that we are only classifying with two classes, fluid and background, so despite having a class imbalance that would weaken the effect of cross entropy loss, the benefit of using dice loss could not be applied.

5.3 Best Model

After conducting experiments on varying batch size, data augmentation, and loss function, the model with the best IoU and Dice score is outlined in Table 5, achieving an IoU of 45.29 and a Dice Score of 62.34. The full training pipeline can be found at `https://github.com/ACK101101/661-DME-Transfer-Learning`.

Table 5: Final pipeline of best performing model

Stage	Parameter	Value
Model	Decoder Head	PSPNet
	Pre-Train Dataset	CHASE
Data Preprocessing	Normalization	μ=46.6002, σ=54.681
Data Augmentation	Random Crop	size=256x256
	Random Resize	ratio range=(0.5, 2.0)
	Random Flip	prob=0.5
Training	Batch Size	8
	Learning Rate	0.01
	Loss Function	Cross Entropy

6 Difficulties

In the initial setup of our working environments, we faced issues with building and running any model using the OpenMMLab libraries. Some of us couldn't run Jupyter Notebooks but could run raw Python files that used the MMSegmentation library, while others had the opposite issue. This occurred despite having the same library versions, routes of installation, and operating system. This problem was not solved entirely, but everyone was eventually able to run and train models in some capacity by tweaking installations.

Another large difficulty, in the beginning, was getting the model to train within a reasonable number of training iterations. Initially, our models took 10,000 training epochs with a batch size of 4 to get non-zero results with IoU and accuracy scores in the single digits. Eventually, this was resolved through trial-and-error of tweaking every configuration parameter until the model began training and reaching convergence within a few hundred training epochs.

One difficulty that impacted our progress throughout the project was the lack of documentation for MMSegmentation. While the Tutorial guides helped set up an initial project, there was not much documentation on what keys and what options were available for specific keys of the configuration files. Things like the types of losses, performance metrics, etc. available were not immediately apparent and we needed to go through the source code to find our available options.

7 Future Works

As demonstrated by our results, many pre-trained base models are good candidates for finetuning on the DME task using transfer learning. Surprisingly, models pre-trained on Cityscapes performed very well. This marks an area of future exploration, emphasizing the importance of pretraining on a massive dataset. Because of the relatively smaller sizes of the models pre-trained on medical datasets, we propose exploring ensemble methods. This could theoretically capture valuable pre-trained features across the various candidate base models. We are also identifying a need to analyze which type and level of features are maintained after the finetuning process. This would provide greater insight into the transfer learning process, recognizing which features are valuable for different downstream tasks. We are particularly interested in using this to explore which Cityscape features are valuable for the DME specific task.

8 Conclusion

Through this project, we have successfully demonstrated the capability of adapting an image segmentation model pre-trained on datasets of similar domains towards a more specific task through transfer learning. Moreover, we further explored the potential of the models by fine-tuning the hyperparameters and tested different loss functions, achieving an IoU of 45.29% and a Dice score of 62.34% on the DME dataset. These metrics highlights the model's enhanced performance in segmenting medical images, demonstrating the effectiveness of our methodology in optimizing model performance for specialized tasks such as medical imaging.

References

[1] MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. 2020, https://github.com/open-mmlab/mmsegmentation.

[2] Ronneberger, Olaf, et al. 'U-Net: Convolutional Networks for Biomedical Image Segmentation'. arXiv [Cs.CV], 2015, http://arxiv.org/abs/1505.04597. arXiv.

[3] Zhao, Hengshuang, et al. 'Pyramid Scene Parsing Network'. arXiv [Cs.CV], 2017, http://arxiv.org/abs/1612.01105. arXiv.

[4] Chen, Liang-Chieh, et al. 'Rethinking Atrous Convolution for Semantic Image Segmentation'. arXiv [Cs.CV], 2017, http://arxiv.org/abs/1706.05587. arXiv.

[5] Long, Jonathan, et al. 'Fully Convolutional Networks for Semantic Segmentation'. arXiv [Cs.CV], 2015, http://arxiv.org/abs/1411.4038. arXiv.

[6] Chiu, Stephanie J., et al. 'Kernel Regression Based Segmentation of Optical Coherence Tomography Images with Diabetic Macular Edema'. Biomed. Opt. Express, vol. 6, no. 4, Optica Publishing Group, Apr. 2015, pp. 1172–1194, https://doi.org10.1364/BOE.6.001172.

[7] M. M. Fraz et al., 'An Ensemble Classification-Based Approach Applied to Retinal Blood Vessel Segmentation,' in IEEE Transactions on Biomedical Engineering, vol. 59, no. 9, pp. 2538-2548, Sept. 2012, doi: 10.1109/TBME.2012.2205687.

[8] Staal, J., et al. 'Ridge-Based Vessel Segmentation in Color Images of the Retina'. IEEE Transactions on Medical Imaging, vol. 23, no. 4, 2004, pp. 501–509, https://doi.org10.1109/TMI.2004.825627.

[9] Budai, A et al. "Robust vessel segmentation in fundus images." International journal of biomedical imaging vol. 2013 (2013): 154860. doi:10.1155/2013/154860

[10] A. Hoover and M. Goldbaum, "Locating the optic nerve in a retinal image using the fuzzy convergence of the blood vessels," in IEEE Transactions on Medical Imaging, vol. 22, no. 8, pp. 951-958, Aug. 2003, doi: 10.1109/TMI.2003.815900.

[11] Zoetmulder, R., Gavves, E., Caan, M., & Marquering, H. (2022). Domain- and task-specific transfer learning for medical segmentation tasks. Computer Methods and Programs in Biomedicine, 214, 106539. https://doi.org/10.1016/j.cmpb.2021.106539

[12] Yukun Guo, Tristan T. Hormel, Honglian Xiong, Jie Wang, Thomas S. Hwang, Yali Jia; Automated Segmentation of Retinal Fluid Volumes From Structural and Angiographic Optical Coherence Tomography Using Deep Learning. Trans. Vis. Sci. Tech. 2020;9(2):54. https://doi.org/10.1167/tvst.9.2.54.

[13] Kolokythas, Antonia, et al. 'Use of Artificial Intelligence in the Characterization of Oral Mucosal Lesions'. Journal of Oral and Maxillofacial Surgery, vol. 80, no. 9, Supplement, 2022, pp. S3–S4, https://doi.org10.1016/j.joms.2022.07.009.