Image Pretraining Methods for OCT Image Segmentation: A Comparison and Analysis of Supervised and Self-Supervised Methods

Justin Knopfmacher and Richard Klein
School of Computer Science and Applied Mathematics
University of Witwatersrand
Johannesburg, South Africa
justin.knopfmacher1@students.wits.ac.za, richard.klein@wits.ac.za

***Abstract*—Diagnosis of ocular diseases is labour-intensive. Automated segmentation of Optical Coherence Tomography (OCT) scans can help diagnose these diseases by showing the structure and thickness of the retinal layers. However, most segmentation methods rely on costly labelled data. This paper examines the effect of different image pretraining methods on OCT image segmentation tasks. We pretrain the encoder of a U-Net using supervised and self-supervised methods on a large OCT dataset with classification labels. The methods compared include BYOL, PLC, SimCLR, SimCLRv2, DINO, and classification pretraining. Then, we fine-tune the whole U-Net on a smaller OCT dataset with segmentation masks for the downstream segmentation task. We compare the dice scores per layer between the pretrained and baseline models. Most of the pretraining methods provide minor improvements compared to the baseline on a dataset that is very homogeneous, with the best results from PLC pretraining and from pretraining on the labelled classification problem via transfer learning.**

Index Terms—OCT Image Segmentation, Image Pretraining, U-Net, Medical Image Segmentation, Transfer Learning, DINO, SimCLR, SimCLRv2, BYOL, PLC

I. Introduction

Diagnosis of ocular diseases is a labour-intensive task [8]. Therefore, to prevent many patients from going undiagnosed, automated methods have been developed to assist with the diagnosis using Optical Coherence Tomography (OCT) scans [14]. These scans are non-invasive imaging of the cross-section of the retina and contain vital data in diagnosing and monitoring eye diseases [21].

The predominant techniques for analysing OCT scans make use of image segmentation [20]. Labels are assigned to each pixel in an image by separating the image into distinct segments [8]. These labels can then be used to facilitate diagnosis and monitor the progression of disease [14]. This is very laborious and error-prone when done by a human, so an automated approach is often taken [20]. However, automating this task comes with additional challenges such as dealing with noise, artefacts, and retinal structure variability [10]. Therefore, advanced approaches are required to get robust segmentation results.

The most prevalent state-of-the-art techniques for segmentation usually employ deep learning using an approach called semantic segmentation [21]. Deep learning is a technique that learns key features from data over time. Deep learning approaches to segmentation are effective at learning complex features from images and can mostly deal well with issues like noise and artefacts [18]. The most significant constraints of these state-of-the-art models are the necessity for extensive, annotated datasets, the incapacity to handle entirely novel data, and the sub-optimal performance on low-quality OCT images [20, 17]. Therefore, alternative methods are examined to address these limitations and improve the generalization and robustness of segmentation models.

The main objective of this research is to address the challenge of the high cost of labelled data. Labelled data requires a lot of human effort and resources, as well as highly skilled experts who can annotate images with pixel-level accuracy. Most modern state-of-the-art models rely heavily on labelled data only. We evaluate various state-of-the-art pretraining methods to demonstrate how leveraging large volumes of unsegmented data enhances the performance of the target task.

The rest of this paper is organized as follows: Section II gives a background on semantic segmentation of OCT images and the various definitions. Section III reviews the current state-of-the-art techniques for medical image analysis and identifies the gaps we aim to fill. Section IV describes our methodology, which involves image pretraining and U-Net fine-tuning. Section V presents and analyzes our results, and compares them with a baseline model. Section VI discusses the limitations of our approach and suggests some future research directions. Section VII concludes the paper and summarizes the main contributions.

II. Background

This section will discuss a background for the basics of this research to understand the research problem and its significance. We cover the following topics: Section II-A introduces the retinal scans we analyze. The application of semantic segmentation in the OCT domain is discussed in Section II-B. Finally, the U-Net, which is the primary model used throughout this paper, is introduced in Section II-C.

A. OCT Imaging

OCT is an acronym for Optical Coherence Tomography, which is a non-invasive technique to obtain cross-sectional images of biological tissues. OCT images are captured by measuring the time delay and intensity of light that are reflected by different depths inside a tissue which allows the construction of a depth profile. OCT images are particularly useful since they often have high resolution, high contrast and require no special preparation to capture [14]. OCT retinal scans can show the structure and thickness of retinal layers. These are important features for assessing health and tracking disease progression [8].

B. Semantic Segmentation of OCT Scans

Semantic segmentation of OCT images is the task of assigning a label to each pixel in an OCT image, indicating the tissue type or pathology present. This can enable automated analysis and diagnosis of various ocular diseases by emphasising the separation of the retinal layers [21]. Semantic segmentation of OCT images poses several challenges, such as human variability in retinal structures and the difficulty of segmenting retinal fluid pockets [8, 10]. Therefore, convolutional neural networks are employed to learn rich features and perform pixel-wise classification [21].

C. U-Net for Semantic Segmentation

U-Net is a convolutional neural network that was developed specifically for medical image segmentation [22]. U-Net has a U-shaped architecture that consists of two parts: an encoder and a decoder. The encoder performs feature extraction from the input image, while the decoder performs up-sampling of the encoder's feature map and restores details [18]. U-Net also utilises skip connections between the encoder and the decoder, which help to preserve low-level features and improve the segmentation accuracy [22]. U-Nets are widely used as the baseline model for semantic segmentation, and many variants and extensions have been proposed to enhance their performance [19, 10].

III. Related work

This section surveys the recent literature on medical image segmentation and classification using deep learning methods. We categorize the existing methods into two groups: those that modify the U-Net architecture in Section III-A, and those that pretrain the U-Net encoder using self-supervised or semi-supervised techniques to leverage unlabeled data in Section III-B. Finally, we will discuss how we can contribute to the literature in Section III-C.

A. U-Net Modifications

The U-Net model is often modified to increase its complexity and expressiveness. The original U-Net has one encoder and one decoder branch, but some works have proposed using multiple encoders to capture different types of information from the input images.

For example, Y-net is a modified U-Net with two encoder branches: a spatial encoder that extracts features from the image domain, and a spectral encoder that extracts features from the frequency domain [10]. The two branches are then fused into a single decoder to produce the segmentation output. This method was motivated by the challenge of segmenting intraretinal fluid pockets, which are often noisy and low-contrast.

Another approach also used multiple encoders, called auxiliary encoders, to combine information from two different domains: labelled and unlabeled data [20]. This approach applied a semi-supervised learning framework that alternates between supervised and self-supervised training of the encoders and the decoder.

Both approaches showed significant improvement over the standard U-Net due to the integration of information from multiple domains.

B. Encoder Pretraining

Another group of methods focuses on pretraining the U-Net encoder using self-supervised or semi-supervised techniques, without altering the U-Net architecture. These methods aim to exploit unlabeled data to learn useful features for medical image analysis and reduce the reliance on large amounts of annotated data.

For example, SimCLR, a self-supervised technique that learns features by contrasting different views of the same image [5], was used to pretrain a ResNet-50 encoder for medical image classification [2]. The encoder was first pretrained on ImageNet and then on unlabeled medical images from various domains. The results showed that this pretraining on the medical images improved the accuracy of medical image classifiers, especially when the labelled data was limited. The paper concluded that self-supervised pretraining can be beneficial for medical image classification and that it can potentially improve performance in other domains as well.

Additionally, a self-supervised pretraining method based on BYOL, another technique that learns features by contrasting different views of the same image, was proposed for medical image analysis [15]. BYOL used two U-Nets, an online one and a target one, that tried to predict each other's output given different views of the same unlabeled image. The target network was updated using a moving average of the online network weights. BYOL was pretrained on multiple domains, including ImageNet and cardiac segmentation datasets. The pretraining significantly improved the data efficiency and performance of the segmentation model. Fine-tuning on a only few labelled samples after pretraining produced robust segmentation results. This shows the effectiveness of this pretraining method for semantic segmentation.

Both of the above approaches showed that pretraining a U-Net encoder on unlabeled data could significantly improve the performance of the downstream task.

C. Our contribution

The majority of the existing work on OCT image segmentation using deep learning methods has concentrated on altering the U-Net architecture to improve its complexity and expressiveness. Pretraining methods have been employed for other types of medical image analysis with outstanding results. However, research is scarce on the effect of pretraining techniques for medical image segmentation, especially for OCT images. Therefore, we decided to investigate a variety of pretraining techniques, both old and new, and see how they can enhance the performance of a standard U-Net in OCT image segmentation.

IV. Methodology

In this section, we describe our methodology for OCT image segmentation. The methodology is divided into three main parts: the data, the pretraining, and the fine-tuning. Section IV-A covers the sources, formats, and preprocessing of the OCT images. Section IV-B explains the different methods we used to pretrain our segmentation model on unlabeled data. Then IV-C discusses how we fine-tuned our pretrained model on the labelled data and evaluated its performance. The code for our entire methodology is available on a Github repository[1].

A. Datasets

This research relies on two datasets: a categorisation dataset for pretraining and a segmentation dataset for fine-tuning. To reduce computational costs, all images are resized to 128 by 128 pixels. To maximise compatibility with existing libraries and GitHub repositories, all images are converted to RGB. Both datasets have no sensitive personally identifiable medical data and are freely available for academic research.

1) Classification Dataset: The dataset used for pretraining is the OCT2017 dataset, which contains 84 495 Optical Coherence Tomography (OCT) images of the retina [16]. The dataset was created by Kermany et al. from the Shiley Eye Institute, University of California, San Diego, and other collaborators from China, Germany, and the USA. The images are labelled into four categories: NORMAL, CNV, DME, and DRUSEN, which correspond to different retinal diseases. We follow the data split provided by the dataset authors to avoid any data leakage or duplication. This dataset is used to pretrain the encoder to learn how to encode the OCT images effectively. We used the labels for one pretraining method, where we trained a ResNet model on image classification with a random sample of 10000 training images and the whole test and validation sets. For all other pretraining methods, such as DINO and BYOL, we pretrained the model on self-supervised learning tasks with the labels discarded and only using the training set. Therefore, the dataset is mainly used for the volume and variety of data to learn the structure of the images, rather than for the semantic information of the labels.

[1]https://github.com/J-HyperNova/OCT-Pretraining

2) Segmentation Dataset: The Chiu BOE 2015 dataset from Duke University [7] was used to fine-tune our segmentation model. This dataset contains 110 B-scans from ten patients with severe DME pathology. The original dataset was highly compressed, so we applied the publicly available code from the Y-net paper [10] to decompress the images and split them into train, test, and validation sets, ensuring that no patient was repeated across the sets. Each image had a corresponding segmentation mask with 11 labels, each representing a different part of the retina or the background. This is the dataset that we evaluate the performance of our models. However, this dataset has a limitation of being very homogeneous, as all the scans are from only 10 patients, captured by the same device with the same disease at the same level of progression. This will affect how we interpret the results and compare the different methods.

B. Pretraining Methods

This paper explores the potential of pretraining on unlabeled data to enhance the performance of a model when labelled data is limited. Pretraining is a technique to learn general and useful representations from unlabeled data that can be fine-tuned on labelled data. We used a randomly initialised ResNet-50 model as the encoder of a U-Net segmentation model and pretrained its weights using different methods. We used the Adam optimizer for all the pretraining methods. We relied on existing code on GitHub for implementing the pretraining methods and used the default hyperparameters provided by the code unless otherwise specified. All the models were built on top of PyTorch and Torchvision. We catalogued the hyperparameters and training conditions in Table I.

1) BYOL which stands for Bootstrap Your Own Latent, is a state-of-the-art technique for learning image features without using any labels [15]. It works by creating two different views of the same image and training two U-Nets, called the online and the target networks, to contrast them. The online network learns to produce the same output as the target network, which is slowly updated by copying the online network weights with a moving average, which is the bootstrapping component. A contrastive loss function is used to measure the similarity between the two networks. The ResNet-50 weights saved for the downstream task come from the online network.

2) Classification pretraining involved training a ResNet-50 model on the OCT2017 dataset while utilising the given classification labels. We initialised the ResNet weights to the ImageNet pretraining and then modified the output layer of the ResNet model to have four nodes, corresponding to the four classes. We used only 10000 images from the training set and tested the model on the test set. The model achieved an impressive F1 score of 0.98348 on the test set, demonstrating that it learned the classification task well with a small sample size. We then used this entire model as a pre-trained encoder for our segmentation task, by restoring the original output layer. This is a form of transfer learning with the hope that the features learnt for classification translate well to the segmentation problem.

3) DINO is a self-supervised learning method for training computer vision models on unlabeled images [4]. DINO stands for Distillation with NO labels, and it uses two networks, a student and a teacher, to learn features from different transformations or augmentations of the same image. The student network tries to predict the output of the teacher network, which is updated by a moving average of the student network weights. The student and the teacher networks are trained by minimizing the cross-entropy loss between their outputs. The student network's encoder weights are used for the downstream task.

4) PLC is a method of learning visual features from unlabeled images by using pixel-level pretext tasks [25]. The acronym PLC stands for Pixel-Level Consistency, which reflects the goal of learning features that are consistent across different images and regions. PLC is designed for downstream tasks that require dense pixel predictions, such as object detection and semantic segmentation. The framework has two components: a backbone convolutional neural network (CNN) and a head vision transformer (ViT). The CNN extracts features from the input images, and the ViT converts them into pixel embeddings. The framework is trained on two pixel-level pretext tasks: pixel-level contrastive learning and pixel-to-propagation consistency. The pixel-level contrastive learning task encourages the pixel embeddings to be similar for the same image and different for different images. The pixel-to-propagation consistency task enforces the pixel embeddings to be coherent with the propagated embeddings, which are obtained by applying a dense conditional random field (CRF) model to smooth the pixel embeddings. By training on these tasks, PLC aims to learn features that are invariant to image transformations, discriminative for image recognition, coherent across image regions, and meaningful for pixel-level prediction. The encoder weights learnt in the backbone network are saved for the segmentation task.

5) SimCLR is a self-supervised learning method for training computer vision models on unlabeled images [5]. SimCLR stands for Simple Contrastive Learning of Visual Representations, and it uses two networks, an encoder and a projection head, to learn features from different augmentations of the same image. The encoder network maps the input image to a latent representation, and the projection head maps the latent representation to a contrastive embedding. The contrastive loss maximizes the similarity between embeddings of the same image under different augmentations and minimizes the similarity between embeddings of different images. The ResNet-50 weights saved for the downstream task come directly from the encoder network.

6) SimCLRv2 is an extension of SimCLR and is considered the state-of-the-art. SimCLRv2 introduces several enhancements to SimCLR [6], such as:

- A deeper projection head with non-linearities that maps the latent representations to contrastive embeddings. This improves the quality and diversity of the embeddings.
- A memory mechanism that stores the outputs of a teacher network as negative samples. This increases the contrastive loss and reduces the dependence on large batch sizes.

SimCLRv2 learns features that are useful for image segmentation by using different augmentations of the same image as positive pairs, and different images or teacher network outputs as negative pairs. SimCLRv2 outperforms SimCLR on image classification by a large margin [6], especially when using a few labelled examples for fine-tuning. The ResNet-50 weights saved for the downstream task are directly from the encoder network.

7) ImageNet is a dataset of natural images that covers 1000 different categories and has more than 1.2 million images [9]. Many image analysis models use these pre-trained weights, as it saves time and effort for learning from data. However, ImageNet images are mostly natural images of objects, animals, plants and scenes. They are very different from medical images, which are usually grayscale, low-resolution, and contain specific anatomical structures or pathological features. Therefore, using ImageNet weights may not help our downstream task performance and may even degrade them by introducing irrelevant or misleading information. The ResNet-50 weights for the downstream task are directly imported from PyTorch.

C. Fine-Tuning

This is the main focus of the research paper and is the downstream task on which the pretraining performance is evaluated.

1) Training: To fine-tune the encoders for a segmentation task, we used a standard U-Net model from a package called Segmentation Models PyTorch [13], which provides various segmentation architectures and pre-trained encoders. We set the number of classes to 11, following the original dataset annotation, but we ignored the 11th class in our analysis, as it was not used by previous works [10, 11]. We initialized the encoder weights with the pre-trained ResNet-50 models from our self-supervised and supervised learning methods. We used Weights & Biases [3] to conduct a hyperparameter sweep and find the optimal settings for each model. The hyperparameters that we swept were: total epochs, learning rate, batch size, weight decay, and early stopping patience. We selected the best hyperparameters based on the lowest validation loss. The optimal hyperparameters used are included in Table II. Weights & biases was also used to track the training process and produce the validation loss in Figure 1.

2) Loss: We adopted a combination of dice loss and focal loss as our loss function for the segmentation task, inspired by the recent works that suggested this loss combination for dealing with class imbalance in medical image segmentation [26, 27]. These works demonstrated that this loss combination can achieve a better trade-off between precision and recall, compared to other loss functions such as cross-entropy, dice, or Tversky.

TABLE I: Details about the pretraining process

Method	Epochs	LR	Batch Size	Proj Dim	Sim Temp	Code	GPU	Train Time
BYOL	50	3e-4	320	N/A	N/A	BYOL[24]	1 x RTX3090	9h
Class	50	1e-4	32	N/A	N/A	Torchvision	2 x GTX1060	5h
DINO	50	1e-4	128	256	0.3	PySSL [12]	1 x RTX3090	11h
PLC	50	1e-4	256	256	0.3	PLC [23]	1 x RTX3090	10h
SimCLR	50	3e-4	1024	256	N/A	PySSL [12]	2 x Quadro RTX 8000	18h
SimCLRv2	50	1e-4	128	128	N/A	PySSL [12]	1 x RTX3090	12h

TABLE II: Hyper parameters used when fine tuning on the segmentation task

Weights	Epochs	LR	Batch Size	Weight Decay	Patience
BYOL	40	7.998e-4	2	1e-5	15
Class	50	6.981e-4	2	1e-5	15
DINO	50	6.4659e-4	2	1e-5	15
PLC	40	8.608e-4	2	1e-5	15
SimCLR	50	8.008e-4	2	1e-5	15
SimCLRv2	100	8.796e-4	2	1e-5	15
ImageNet	150	9.9956e-4	2	1e-4	15
Random	150	7.509e-4	2	1e-5	15

Dice loss is a common similarity measure between two sets of binary labels, which is widely used for evaluating segmentation models. Dice loss is computed as:

$$\text{Dice loss} = 1 - \frac{2|A \cap B|}{|A| + |B|},$$

where A and B are the sets of predicted and ground truth pixels, respectively. Dice loss is zero when the sets are identical, and one when they are disjoint. Dice loss can handle class imbalance by giving more importance to the overlap of the sets, rather than the individual sizes.

However, for multi-class segmentation problems, where each pixel can belong to one of several classes, the binary dice loss is not sufficient. Therefore, we used the macro-average dice loss from the Segmentation Models PyTorch package [13], which calculates the dice loss for each class separately and then averages them over all the classes. The formula is:

$$\text{Macro-average dice loss} = 1 - \frac{1}{C}\sum_{c=1}^{C}\frac{2|A_c \cap B_c|}{|A_c| + |B_c|},$$

where C is the number of classes, A_c and B_c are the sets of predicted and ground truth pixels for class c, respectively. This loss gives equal weight to each class, regardless of its frequency or size.

Focal loss is a modification of cross-entropy loss that concentrates on hard examples and diminishes the contribution of easy examples. Focal loss is calculated as:

$$\text{Focal loss} = -\alpha_t(1 - p_t)^\gamma \log p_t,$$

where p_t is the predicted probability of the true class, α_t is a weighting factor for the true class, and γ is a focusing parameter that adjusts how much the easy examples are down-weighted. Focal loss can handle class imbalance by assigning higher weights to rare classes and lower weights to frequent classes, as well as modulating the loss by the prediction confidence.

By combining dice loss and focal loss, the benefits of both methods can be exploited, improving the performance of our segmentation model. Dice loss measures the global similarity between the predicted and ground truth masks, while focal loss adjusts the loss by the prediction confidence and the class frequency. This way, both spatial and class imbalance in the segmentation task can be handled. We computed our final loss as follows:

$$\text{Total loss} = \text{Macro-average dice loss} + \text{Focal loss}.$$

V. Analysis of Results

In this section, we present the results of our experiments using different pre-training methods. Our labelled segmentation dataset is very limited, which motivated us to use pretraining in the first place. Therefore, using a single metric to compare different pretraining methods may not capture the whole picture. The baseline model, where the encoder weights are randomly initialised, already achieves excellent results on this task, due to the homogeneity of the dataset, the efficiency of the U-Net structure, and the effectiveness of using the dice-focal loss for segmentation. However, some pretraining methods can still provide small but noticeable improvements. From now on, U-Net models will be referred to by the pretraining technique used to initialize their encoders. "Random" will be used to refer to the previously defined baseline model. We divide our analysis into multiple sections.

Firstly, we examine the differences in the pretraining and fine-tuning process for each method in Sections V-A and V-B. We also show the validation loss curves of the models during training and discuss their convergence speed and stability in Section V-C. Next in Section V-D, we conduct a qualitative analysis by showing the segmentation masks of a sample OCT image generated by each model and visually assessing their quality and consistency. Then, we explain and justify the use of the dice score as our evaluation metric in Section V-E which is used in our final comparison between pretraining methods in Section V-F. Finally, we provide a holistic discussion of the advantages and disadvantages of each method in Section V-G.

A. Pretraining Time and Compute Cost

The pretraining methods we used varied in their time and compute requirements, as shown in Table I. The classification pretraining was the fastest and least computationally demanding method, but it also depended on the labels of the classification dataset, which made it different from the other methods that were all self-supervised. The slowest and most resource-intensive method was SimCLR. This method took the longest time to train and needed two expensive workstation-grade GPUs to handle the large batch size required. We will evaluate the downstream performance to see if this extra computation cost is justified, especially when SimCLRv2 is available. The other methods, BYOL, PLC and DINO, were more moderate in both time and GPU usage, with BYOL being the quickest of the self-supervised methods. Using ImageNet as a pretraining method was the easiest option, as encoders pretrained on ImageNet are already available, but we will show later that more domain-specific pretraining is much more beneficial. It should be noted that the hyperparameters during pretraining were fixed and were not experimented with to further optimise the results due to resource constraints.

B. Optimal Segmentation Hyperparameters

To compare the effects of different pre-trained weights on the fine-tuning of the segmentation U-Net, we conducted a sweep to find the optimal hyperparameters for each case. We observed some interesting patterns in the final optimal hyperparameters in Table II:

- All of the pre-trained models, except for SimCLRv2, ran for a maximum of 50 epochs, whereas random and ImageNet required around 150 epochs to converge to their optimal validation loss.
- All of the cases converged to a batch size of 2, which indicates that a small batch size is beneficial for avoiding overfitting and improving generalization on this task. A small batch size also allows for more gradient updates per epoch, which can speed up the convergence.
- ImageNet had both the highest learning rate and the highest weight decay, which implies that these weights are not very helpful for medical image segmentation. This possibly indicates that a large drift from the original weights was required to obtain decent results in this domain.
- All models converged to a patience of 15 for their early stopping, which shows that sometimes a small worsening of validation loss initially can be overcome by further training. A patience of 15 also balances between stopping too early and wasting computational resources.
- DINO and the classification model both had the smallest learning rate, which possibly suggests that the weights learnt did not need to change a lot to obtain a low validation loss.

C. Validation Loss

All models converge to similar final validation loss scores when their optimal hyperparameters are used. However, their patterns of convergence and final stopping points give significant insights into the features learnt from pretraining. It can be observed from the validation loss plot in Figure 1 that the following trends are present:

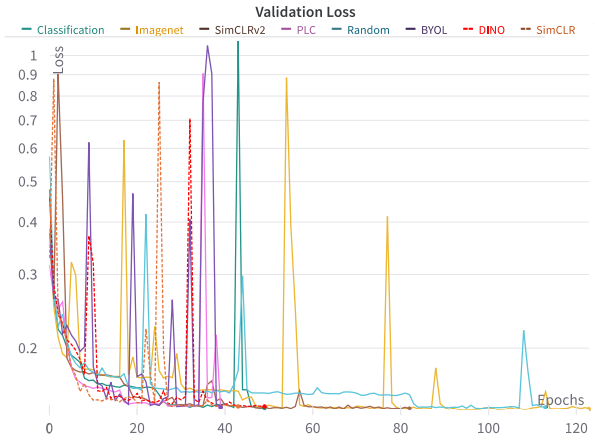


Fig. 1: Validation Loss for all weights, log scale y-axis

- All of our pretrained weights converged faster than the baseline of random weights.
- For the baseline of random weights, we notice that it converged to a much higher validation loss than all the other methods until around 80 epochs when it dropped down.
- ImageNet had the worst performance, as it converged to a much higher validation loss and stopped at the highest epoch count of over 120. This could show that significant fine-tuning was done to adapt to the segmentation task.
- SimCLRv2 took longer than the other pretraining methods to converge to the lowest validation loss and stopped around 80 epochs. This may imply that SimCLRv2 also required more fine-tuning to adjust to the new task, however, the final result may still be notable.
- BYOL and PLC converged the fastest, both stopping around the same number of epochs. This might show that both methods learnt very effective encodings.
- The spikes in the validation loss are caused by the model taking a step in a certain direction that improved its fit to the training data but worsened its fit to the validation data. However, since the spikes all recovered after a few epochs, they did not significantly affect the performance. The constant spikes could be caused by the very small batch size, which makes the gradient estimates noisy and unstable. However, the sweeps proved that this smaller batch size produces the best validation loss overall.

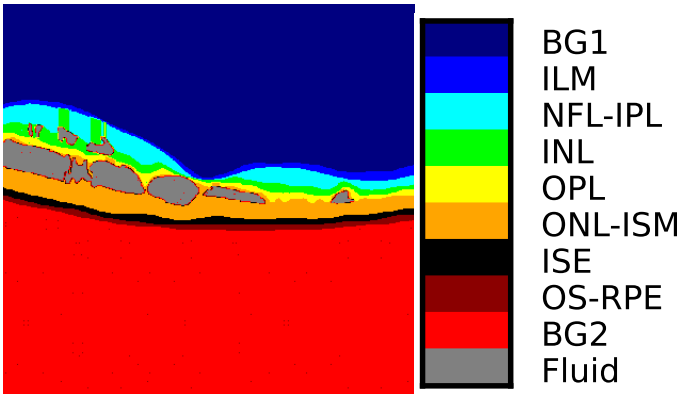
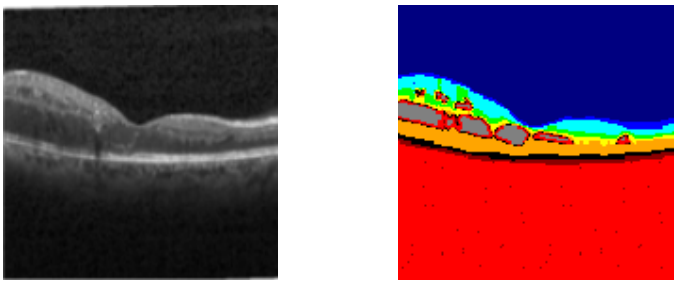
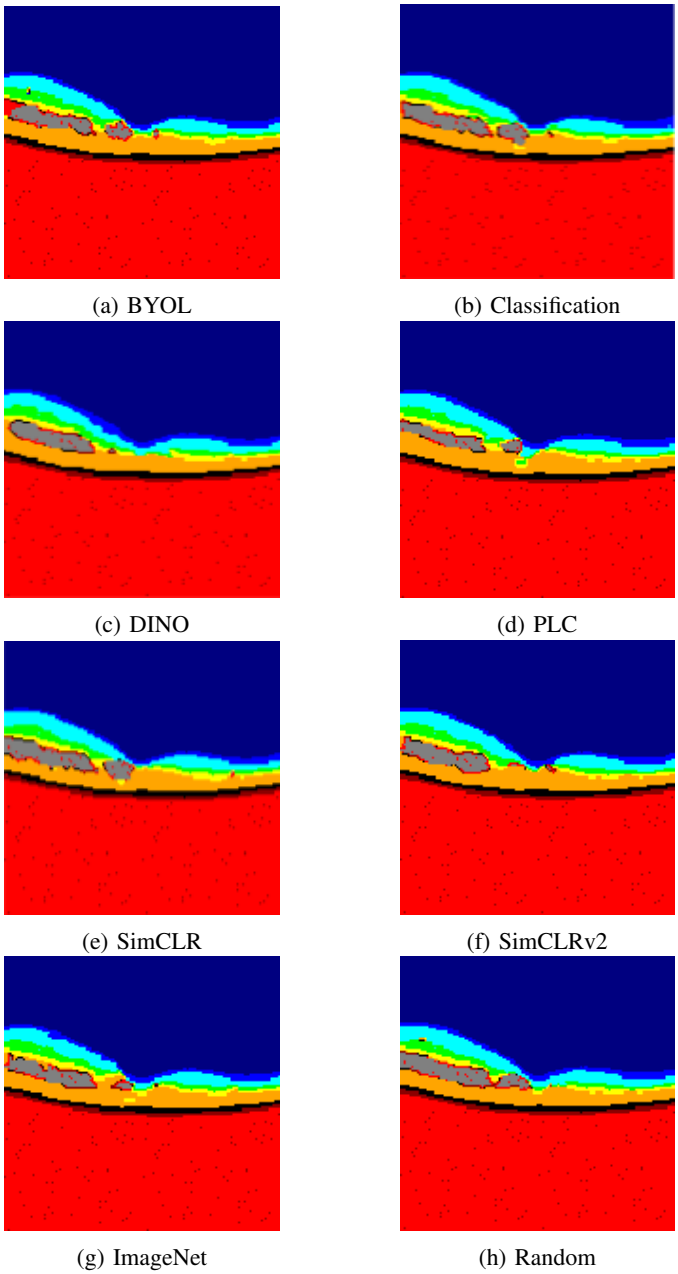


Fig. 2: Legend for each retinal layer and their corresponding colour at the original resolution



(a) Original OCT image (b) Original Segmentation

Fig. 3: Sample OCT image and the corresponding segmentation in the reduced resolution



(a) BYOL (b) Classification

(c) DINO (d) PLC

(e) SimCLR (f) SimCLRv2

(g) ImageNet (h) Random

Fig. 4: Predicted segmentations by each pretrained U-Net

D. Qualitative Analysis

The segmentation of the OCT image aims to separate each retinal layer. Each layer can provide information about eye health, both individually and collectively. Therefore, it is important to evaluate the performance of each pretraining method on each layer separately, rather than relying on an overall accuracy score, even a weighted one. This is because some layers may have more clinical relevance than others, depending on the disease and the purpose of the segmentation.

Figure 2 shows a sample mask from the dataset and its legend, indicating the colour-coded retinal layers, where BG1 represents the upper background, BG2 represents the lower background and FLUID represents the fluid pockets. This helps to interpret the results of the predicted segmentation masks.

In this section, for ease of comparison, we have selected a single difficult-to-segment OCT image and will analyse how well each pretraining method gets close to the ground truth shown in Figure 3.

The segmentation quality is affected by the low image resolution of 128 by 128 pixels, which reduces many details to a few pixels wide. This makes it difficult for the model to learn and recognize these features. However, this resolution was chosen due to time and resource constraints.

- The random baseline (Figure 4h) achieves surprisingly high accuracy, leaving little room for improvement except on very fine details.
- None of the pretraining methods can accurately capture the small fluid pockets between the INL and NFL-IPL layers, as they are only a few pixels wide. They also miss the rightmost fluid pocket.
- ImageNet (Figure 4g) produces uneven lines in its segmentation, while the other methods create smoother transitions between layers. This is especially visible in the NFL-IPL layer and the shape of the fluid pocket, as well as an isolated patch of the OPL layer inside the ONL-ISM layer.

- SimCLR (Figure 4e) and classification (Figure 4b) also have some pixels of the OPL layer below the fluid pockets in between the ONL-ISM layer, but they are better connected to the correct layer than in ImageNet's segmentation.
- DINO (Figure 4c) and SimCLRv2 (Figure 4f) fail to segment a second distinct fluid pocket, which is worse than random weights and ImageNet, which at least recognize its presence, albeit smaller than it should be.
- PLC (Figure 4d) creates a strange anomaly around the middle of the image, where it has an ONL-ISM layer wrapped around a small pocket of OPL and INL. This does not occur in any of the other predictions.
- BYOL (Figure 4a) and PLC (Figure 4d) compete well with the state-of-the-art DINO and SimCLR, often producing smoother and more accurate results around the fluid pockets and how the layers change. This is especially important considering their lower computation costs compared to DINO and SimCLR.
- BYOL (Figure 4a) incorrectly segments a fluid pocket as a background layer. This could be due to the use of the background colour for the border around the fluid pockets, which became almost indistinguishable from the layers after the image compression. This also affects the segmentation of the fluid pockets by the other models as can be seen by how they all dot the fluid pockets with stray background pixels, but none of the others fill an entire fluid pocket with the background like BYOL does.
- SimCLRv2 (Figure 4f), PLC (Figure 4d) and random (Figure 4h) can better capture the potentially jagged edges of the ONL-ISM layer, whereas the other methods tend to flatten it out or randomly indent it.

This example is a particularly challenging sample from the test set, so it may not reflect the general performance of the pretraining methods. Therefore, we will examine the average dice scores per layer over the entire test set, to get a more holistic picture.

E. Evaluation Metric

Following the approach of 2 papers that used the same dataset [10, 11], we computed the dice scores for each layer separately, as well as for the upper and lower backgrounds and the fluid pockets. We also removed label 10 from the results, as it was not used by either paper. We use the per-layer dice score to measure the overlap between the predicted and the real masks for each layer individually. The per-layer dice score is a binary dice score that is the complement of the dice loss used earlier. Dice score is defined as:

$$\text{Dice Score} = \frac{2|A \cap B| + 1}{|A| + |B| + 1},$$

where A and B are the sets of pixels belonging to the predicted and real samples, respectively. The smoothing term of 1 is added to prevent division by zero and to increase the stability of the score. This may slightly inflate the values of the dice score, but the comparison among different methods is still valid, as we use the same smoothing for all of them.

F. Per layer Analysis

Table III compares the performance of the baseline U-Net model with random weights initialization and the pretrained U-Net models with different weights on the OCT image segmentation task. The performance is measured by the dice scores for each layer of the retina on the images in the test set. We note the following:

- Classification pretraining achieves the best overall improvements to the baseline, obtaining the highest dice scores in most of the layers. It achieves the highest scores in ILM, OPL, and ONL-ISM, with the remainder of the layers all being within 0.010 of the best score found.
- BYOL consistently improves over the random weights baseline but never produces the best results for any layer.
- PLC scores slightly higher than BYOL in nearly every category and obtains the best dice scores for the INL and OS-RPE layers.
- SimCLR performs terribly in segmenting the OPL layer but achieves the best dice score for segmenting the fluid pockets. Segmenting fluid pockets is challenging as they can be mistaken for blood vessels, shadows, or noise. However, they are crucial in indicating treatment response. [10]. This makes SimCLR's result notable.
- SimCLR, however, performs worse than random in ONL-ISM, ISE, and OS-RPE layers showing how these results are unbalanced.
- SimCLRv2 has significant improvements over SimCLR and produces a much more robust and well-rounded result. It produces the best results for the NFL-IPL layer and consistently improves over the random weights baseline.
- DINO sees some decreases in performance compared to the random weights baseline in the ILM, NFL-IPL and INL layers.
- ImageNet produces the worst results when segmenting the fluid pockets, worse than random weights. However, produces the best score for the ISE layer, tied with DINO. The ISE layer is important for detecting age-related macular degeneration [1]. So these results are still notable, although a method that gives a better balance, such as DINO, will provide more usage.

Training the models may also be influenced by the random element, as each epoch shuffles the dataset order. But this would only make slight changes to the final results and our patterns are notable enough to discuss. Ideally, we would train each model multiple times with the optimal hyperparameters and take the average results, but this was not feasible due to computational and temporal constraints.

G. Holistic Evaluation

Based on the above results, we will compare and contrast the advantages and disadvantages of each method and their implications for the segmentation task. The fact that all the pre-training weights except for ImageNet converged faster than random weights indicates that the pre-trained weights captured

TABLE III: Per layer dice score: Green=Best, Red=Substantially worse

Weights	BG1	ILM	NFL-IPL	INL	OPL	ONL-ISM	ISE	OS-RPE	BG2	Fluid
BYOL	0.996	0.830	0.854	0.717	0.701	0.872	0.849	0.832	0.996	0.529
Classification	0.996	0.837	0.863	0.729	0.730	0.888	0.860	0.835	0.996	0.630
DINO	0.996	0.809	0.823	0.654	0.652	0.878	0.863	0.836	0.995	0.541
PLC	0.996	0.835	0.864	0.739	0.721	0.881	0.860	0.837	0.996	0.527
SimCLR	0.995	0.830	0.852	0.682	0.005	0.813	0.852	0.811	0.996	0.635
SimCLRv2	0.996	0.831	0.865	0.723	0.695	0.879	0.862	0.829	0.996	0.567
ImageNet	0.996	0.821	0.846	0.709	0.704	0.847	0.863	0.831	0.996	0.300
Random	0.996	0.820	0.836	0.682	0.647	0.875	0.858	0.827	0.995	0.518

useful features for encoding the OCT images, even though this did not necessarily lead to better performance on the test dataset. To summarize our findings, the following points are presented:

- BYOL: This method is very efficient in terms of computation time and resources, both during pretraining and training. It produces a balanced result that shows improvement over random weights in all layers. Although minor issues with segmenting the fluid pocket in the qualitative analysis were noticed, the dice score for the fluid pockets was still an improvement over the baseline.
- Classification: This method demonstrates the power of transfer learning as a pretraining tool for segmentation. It shows that the features learned from OCT images for a classification problem are also useful for a segmentation problem. Transfer learning is much more computationally efficient than self-supervised methods and produces the most significant improvements over the random baseline in most of the layers, according to the per-layer analysis.
- DINO: This method is considered the leading technique in self-supervised learning, but it shows some performance losses compared to the baseline in some layers, which may suggest that the hyperparameters used for pretraining were not optimal for this domain. It also requires a lot of computational resources that do not compensate for the good performance in only one layer. Therefore, it is not very useful, especially when there are other methods available.
- PLC: This method is very effective in both pretraining and training, and it outperforms BYOL in almost every category. PLC is tailored specifically for segmentation, whereas BYOL is a more general pretraining technique. This is the best-performing self-supervised method, and it should be preferred in this domain when data labels are not available.
- SimCLR: This method is the worst of all the pretraining methods used, in terms of both computational efficiency and final segmentation results. It is possible that the hyperparameters used for pretraining were not optimal for this domain, which resulted in issues such as neglecting the OPL layer as seen in the dice score. This method, however, achieved the best performance on the fluid layer, but this is not very helpful for a segmentation that is otherwise of low quality. The classification method is probably a better option for examining the fluid pockets since the rest of the results are more accurate as well and the differences in their fluid pocket dice scores are minor.
- SimCLRv2: This method shows substantial improvements over SimCLR in both computational efficiency and final segmentation results. Although it performs worse on the fluid layer, the segmentation is more balanced and it even achieved the best score on one layer. Therefore, for this domain, it seems that SimCLRv2 should replace its predecessor.
- ImageNet: These weights perform the worst in almost every category, except for SimCLR. This indicates that the natural images used in the ImageNet dataset do not provide helpful features for medical image segmentation and significantly harm the result instead. The segmentation model has to work hard to undo the embedding weights via the longest epochs, highest learning rate, and highest weight decay in order to obtain a decent result.

Hence, pretraining can still offer significant improvements in this domain, with PLC being the best self-supervised method and classification pretraining being the best method overall.

VI. Limitations and Future Research

This study has some limitations that future research should consider and address. First, the dataset for OCT image segmentation was small and homogeneous, with only 10 patients who had severe DME. This may affect the generalizability and robustness of the results, as the segmentation models may vary in performance on different OCT devices, imaging protocols and disease progress. Moreover, our experiments could not assess how well the pretraining methods learned representations of OCT images with the other diseases in the classification dataset because the segmentation dataset only had patients with DME. Second, the pretraining comparison is limited by the high computational cost and time constraints. We were unable to conduct a hyperparameter search for each of the pretraining methods and were unable to reduce the variability of the downstream results by averaging them over multiple runs. Third, we reduced the image resolution for the pretraining, which may limit the amount of information that can be learned from the image and create an upper-performance limit. The low resolution could also make it easier to achieve decent baseline results due to the lower accuracy required.

To overcome these limitations, we suggest some directions for future research. One direction is to explore the effect of pretraining on other segmentation models besides U-Net, such

as DeepLabv3, SegNet, PSPNet or modified U-Nets that have shown better results, such as Y-Net. These models may benefit more from pretraining, but this needs to be verified empirically. Another direction could be to pretrain the entire U-Net and not just the encoder. A further direction is to use a larger and more diverse dataset that covers different OCT devices, imaging protocols, disease conditions, and angles. This would improve the representativeness and reliability of the results, as well as the applicability of the models to different clinical settings. It would also show how well the pretraining learned information from all four diseases and not only DME. These directions may help to address some of the challenges and limitations of this study and advance the field of OCT image segmentation.

VII. Conclusion

This paper explored the effect of different pretraining methods for OCT image segmentation and found that most of them could slightly improve the segmentation of individual layers over the baseline. This shows the potential of pretraining for enhancing the performance of segmentation models but also the limitations when the dataset is small and homogeneous. It was discovered that PLC pretraining was the best self-supervised method among the ones tested. However, using a small amount of labelled data with transfer learning on a classification problem was not only more computationally efficient than using self-supervised methods but also produced better results. Furthermore, it was observed that using ImageNet weights as a pretraining method was the worst choice for this problem, leading to lower segmentation accuracy and higher computation time than the baseline. This highlights the importance of selecting a suitable pretraining method for the relevant downstream domain and task.

References

[1] Jennifer H Acton et al. "Relationship between retinal layer thickness and the visual field in early age-related macular degeneration". In: *Investigative ophthalmology & visual science* 53.12 (2012), pp. 7618–7624.

[2] Shekoofeh Azizi et al. "Big self-supervised models advance medical image classification". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 3478–3488.

[3] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[4] Mathilde Caron et al. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[5] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[6] Ting Chen et al. "Big Self-Supervised Models are Strong Semi-Supervised Learners". In: *arXiv preprint arXiv:2006.10029* (2020).

[7] Stephanie J. Chiu et al. "Kernel regression based segmentation of optical coherence tomography images with diabetic macular edema". In: *Biomed. Opt. Express* 6.4 (Apr. 2015), pp. 1172–1194. DOI: 10.1364/BOE.6.001172. URL: https://opg.optica.org/boe/abstract.cfm?URI=boe-6-4-1172.

[8] Delia Cabrera DeBuc. "A review of algorithms for segmentation of retinal image data using optical coherence tomography". In: *Image Segmentation* 1 (2011), pp. 15–54.

[9] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[10] Azade Farshad et al. "Y-Net: A Spatiospectral Dual-Encoder Network for Medical Image Segmentation". In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part II*. Springer. 2022, pp. 582–592.

[11] Zeyu Fu et al. "MPG-net: Multi-prediction guided network for segmentation of retinal layers in OCT images". In: *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 1299–1303.

[12] Nikolaos Giakoumoglou and Paschalis Giakoumoglou. *PySSL: A PyTorch implementation of Self-Supervised Learning (SSL) methods*. https://github.com/giakou4/pyssl. 2023.

[13] Pavel Iakubovskii. *Segmentation Models Pytorch*. https://github.com/qubvel/segmentation_models.pytorch. 2019.

[14] Raheleh Kafieh, Hossein Rabbani, and Saeed Kermani. "A review of algorithms for segmentation of optical coherence tomography from retina". In: *Journal of medical signals and sensors* 3.1 (2013), p. 45.

[15] András Kalapos and Bálint Gyires-Tóth. "Self-supervised Pretraining for 2D Medical Image Segmentation". In: *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII*. Springer. 2023, pp. 472–484.

[16] Daniel S Kermany et al. "Identifying medical diagnoses and treatable diseases by image-based deep learning". In: *cell* 172.5 (2018), pp. 1122–1131.

[17] Jiaxuan Li et al. "Multi-scale GCN-assisted two-stage network for joint segmentation of retinal layers and discs in peripapillary OCT images". In: *Biomedical Optics Express* 12.4 (2021), pp. 2204–2220.

[18] Shervin Minaee et al. "Image segmentation using deep learning: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021), pp. 3523–3542.

[19] José Ignacio Orlando et al. "U2-net: A bayesian U-net model with epistemic uncertainty feedback for photoreceptor layer segmentation in pathological oct scans". In:

2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). IEEE. 2019, pp. 1441–1445.

[20] Yassine Ouali, Céline Hudelot, and Myriam Tami. "Semi-supervised semantic segmentation with cross-consistency training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2020, pp. 12674–12684.

[21] Mike Pekala et al. "Deep learning based retinal OCT segmentation". In: *Computers in biology and medicine* 114 (2019), p. 103445.

[22] Nahian Siddique et al. "U-net and its variants for medical image segmentation: A review of theory and applications". In: *Ieee Access* 9 (2021), pp. 82031–82057.

[23] Phil Wang. *Pixel-level Contrastive Learning Implementation*. https://github.com/lucidrains/pixel-level-contrastive-learning. 2021.

[24] Phil Wang. *Usable Implementation of "Bootstrap Your Own Latent" self-supervised learning, from Deepmind, in Pytorch*. https://github.com/lucidrains/byol-pytorch. 2023.

[25] Zhenda Xie et al. *Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning*. 2020. arXiv: 2011.10043 [cs.CV].

[26] Michael Yeung et al. "Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation". In: *Computerized Medical Imaging and Graphics* 95 (2022), p. 102026.

[27] Rongjian Zhao et al. "Rethinking dice loss for medical image segmentation". In: *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2020, pp. 851–860.

Declaration

I, Justin Knopfmacher (student number 2356115), affirm that the honours research project submitted herein is my original work and that I have not previously submitted it, in whole or in part, to any other institution for a degree. I also acknowledge that all sources used in this project have been properly cited and referenced, in accordance with the academic standards of the University of the Witwatersrand. I understand that any form of plagiarism is a serious offence and may result in disciplinary action.

Date: 9 November 2023

Signature: