Retinal Blood Vessel Segmentation

Author - Atharva Ketkar

Institution - University of Wisconsin-Madison

Abstract

This project discusses the segmentation of blood vessels in retinal images which could aid in the detection of various retinal diseases. I used the DRIVE dataset, containing 20 annotated training images and 20 unannotated test images for blood vessel segmentation. I implemented a basic Unet model which was assessed through a 10-fold cross validation process. This model was tested by using a F1 score which resulted in a mean score of 0.861 with the highest single-fold score being 0.898, signifying an extreme effectiveness.

Introduction

Retinal blood vessel segmentation is necessary in the diagnosis, screening, treatment, and evaluation of various cardiovascular and ophthalmologic diseases such as diabetes, hypertension, arteriosclerosis and choroidal neovascularization. I use the DRIVE dataset, which includes 20 annotated training and 20 testing images of retinal maps and extraction of branch points. By identifying these blood vessels, medical

professionals can detect problems earlier and can therefore diagnose these problems earlier.

In order to develop this process, I first realized that the dataset was sparse with data considering that there were only 20 training and testing images. The best way to mitigate this problem was to split the images into 128x128 patches, resulting in around 1100 patches per image and more data. The next thing that I realized was that only the training dataset had annotations and this would be a problem when testing how accurate the model was. So in order to avoid this, I used k-fold cross validation with k being 10, and this meant that part of the data would be used for training and for validation. The model that I used to train was a basic U-net model with 50 epochs to train a satisfactory model.

After training this model, the F1 score that was reported in this project was 0.898, demonstrating the effectiveness of the model. In order to see how good the model was performing, I compared the predicted output for some training images to the corresponding mask, and the results were almost identical.

2. Methods

2.1 Dataset

I used the DRIVE dataset in this project. This dataset includes 20 training images and 20 test images. The training images have annotated masks, but the testing set doesn't. Each image in the dataset and each mask are 584x565 in size. Because the size of the

dataset is pretty small, I decided to split the images into overlapping 128x128 grayscale patches with a stride of 64 pixels, resulting in about 1100 patches per image. Each image was normalized to have a value of [0,1], and I just used the training dataset because the test dataset didn't have expert annotations available. Instead, I used a 10-fold cross validation in order to still have a satisfactory evaluation.

2.2 Basic U-Net Model

I used a standard U-net model with encoding and decoding. The encoder includes two convolutional blocks with filter sizes of 16 and 32. After this, there is batch normalization, ReLU, and then max pooling for downsampling. In the bottleneck layer, there is a 64 layer convolutional block for deeper analysis. After this layer, it gets downsized into a final 1x1 convolution block with sigmoid activation.

2.3 Dropout U-Net Model

In this method, I copied the standard U-net before and added dropout 2d layers to try and reduce overfitting. All other parts of the training were kept the same for fair comparisons. The reason I added this as a model was because I anticipated that overfitting would be a problem because of the small data amount. Even with patching, the patches are extremely correlated, so I predicted that by adding dropout layers, it would help mitigate this.

2.4 Training Optimization Process

I used 10-fold cross validation where each fold trained on 90% of the data and tested on 10% of the data for better evaluation. I used Binary Cross Entropy for the loss function because the values that return will be in between [0,1], which will be easier for the

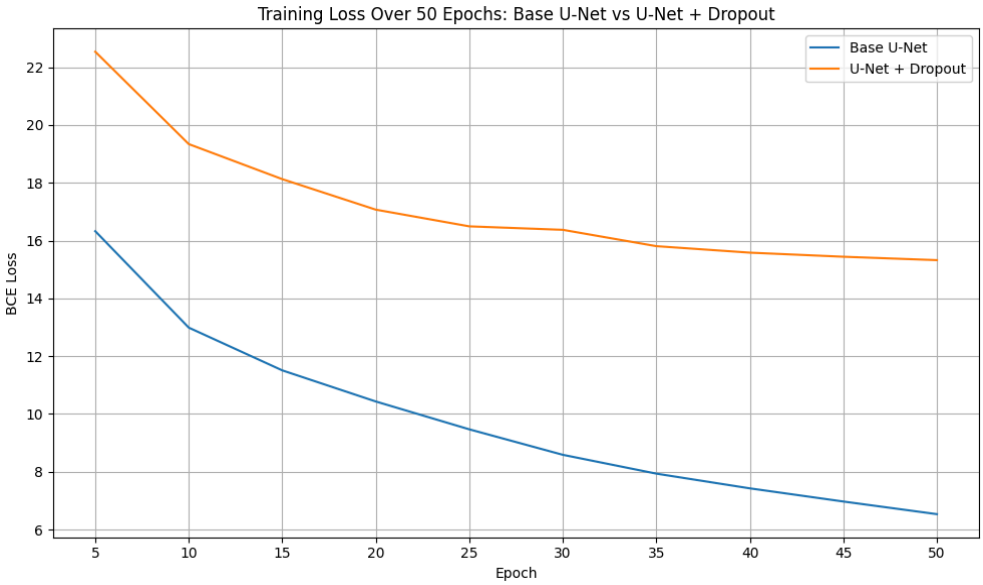
function to identify. I used an Adam optimizer of 0.001 for the model weights which was used in batch sizes of 8, while the model was trained on 50 epochs.

2.5 Metrics

In order to evaluate this project, I used an F1 score. This is because it perfectly balances precision and recall. Precision measures how many predicted vessel pixels are correct, while recall measures how many actual vessel pixels were correctly detected. The F1 score captures both types of error, which is why it would be the best. The non-dropout model was tested by using a F1 score which resulted in a mean score of 0.861 with the highest single-fold score being 0.898, signifying that the methods used worked very well. The dropout model resulted in a F1 mean score of 0.810 with the best single-fold score being 0.860

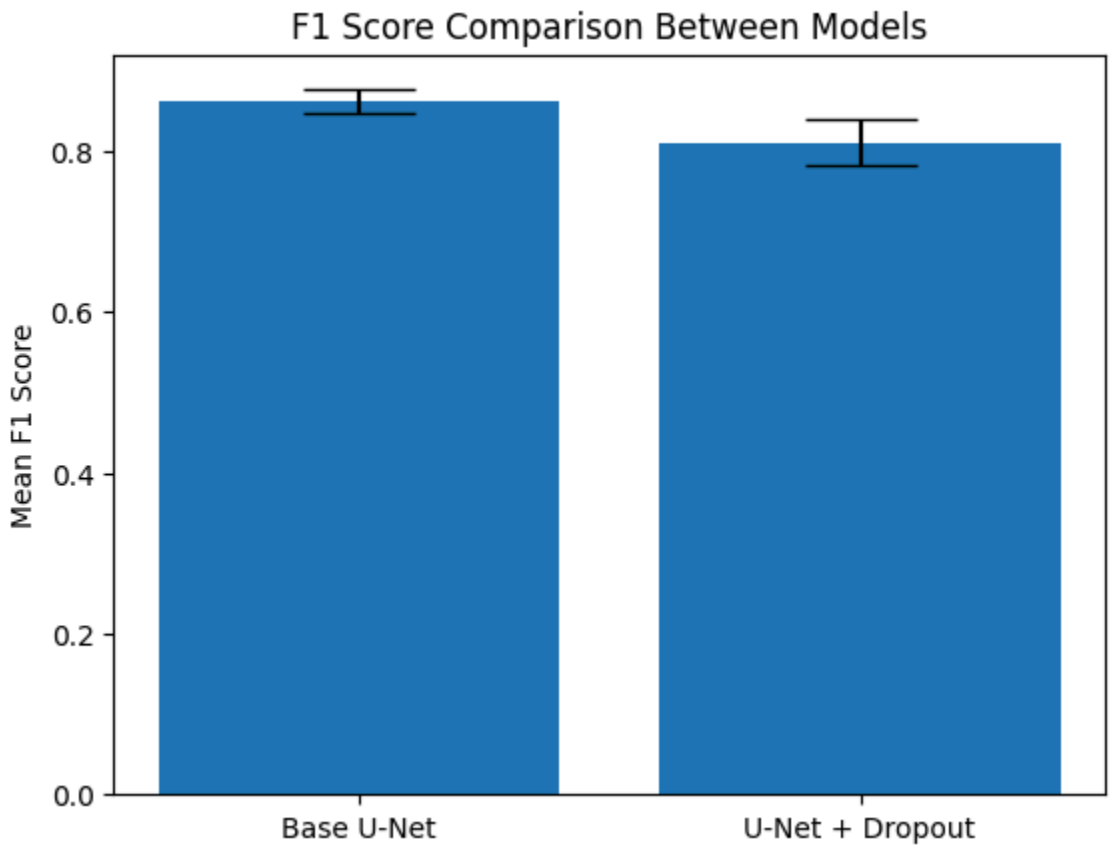
3 Results

3.1 Optimized Method Comparisons



This is a graph showing the training loss over 50 epochs on the best fold for both the methods that I tested in this project: A basic U-net model vs a Dropout model. As is clear, the basic U-net model has lower loss for all epochs, signifying that it performed better than its counterpart.

3.2 Method Result Comparisons



This graph compares the mean F1 score across 10-fold cross-validation for the two models. The base U-Net achieved a mean F1 score of 0.861, while the U-Net with dropout achieved 0.810. The error bars show the standard deviation for each model. The base U-Net model outperformed the dropout variant in average F1 score and had

lower variance, which means it has better consistency. This demonstrates that in this experiment, the simpler baseline model performed better overall on this dataset.

Conclusions

In this experiment, the base model that didn't use dropout was better and received an mean F1 score of 0.861. This means that the more basic and straightforward model was more capable of producing greater results. This might be due to the fact that the training set was already annotated with precision by experts, which reduces a lot of the noise that is present in many other datasets. If I had more time and more resources, I would've used a deeper version of U-net with a bigger size like 128 instead of 64. Also, I would run 100 epochs instead of 50 epochs and could've used more intricate segmentation techniques in order to preprocess the data better.

Limitations and Disclosures

In the DRIVE dataset that I used, there were only 20 training and testing images, but by using patches I alleviated the problems of a small dataset, but not to a full degree. Because I don't have a GPU, I had to use Google Colab's free GPU for this project which gave some computational issues.

I used ChatGPT to learn more about Unet models and debug some code sparsely throughout the project. I didn't get help from anybody or use anyone else's code for this project.

Dataset

https://drive.grand-challenge.org/