Deep Learning Methods to Identify Glaucoma in Optical Coherence Tomography (OCT)

Mikey Joyce
University of Missouri
Department of Electrical Engineering and Computer Science
Columbia, MO

Dr. Filiz Bunyak
University of Missouri
Department of Electrical Engineering and Computer Science
Columbia, MO

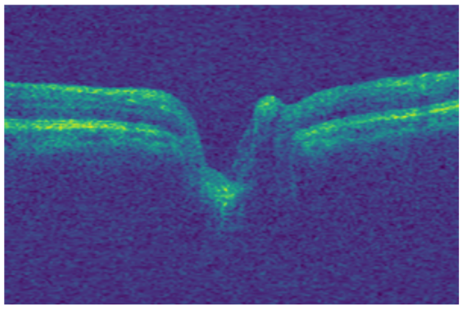
I. Introduction: Motivation and Background

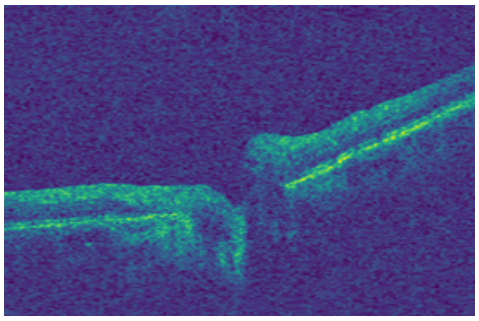
Glaucoma is a complex disease. With more than 3 million Americans suffering from the disease, it is the second leading cause of blindness. Because of this, it is an important field of research in computing to figure out ways to assist ophthalmologists in identifying glaucoma and/or patients who may be at risk for developing glaucoma. Optical Coherence Tomography (OCT) is a type of non-invasive imaging of the retina, which allows ophthalmologists to view the thicknesses of various parts of the retina. Smaller thicknesses of the retina typically are associated with glaucoma. Not only this, but my research revolves around clinical analysis of glaucoma patients, and often metrics we are utilizing are derived from OCT scans, so it is important for me to understand them. It is my goal to review state of the art deep learning methods to analyze OCT images for the classification of glaucoma.

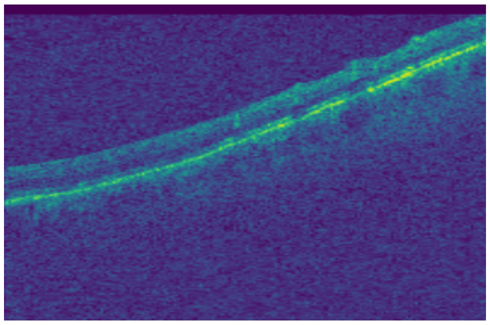
II. Image Processing Data, Modules and Pipeline

A. *The Harvard Glaucoma Detection and Progression Dataset (Harvard GDP) [1]*

The dataset that I am utilizing was curated by the Harvard Ophthalmology AI Lab. It consists of 1000 patients, some of which are just healthy eyes, and other have glaucoma with varying levels of progression. Each patient has an OCT scan which consists of 200 grayscale images with (200, 300) resolution. That means this data set contains 200k images. There exists lots of variance within images in the same scan, and lots of variance between different patients. The variances within the same scan are typically more about the contents of the scan itself. For instance, some parts of the scan show more structured curvature in the retina, while other parts of the scan have just straight lines in the retina. The variance between different patients is typically dealing with the rotations of the image, as they seem to not be centered in the same spot. Some of this variance is also with the different levels of brightness in the images.

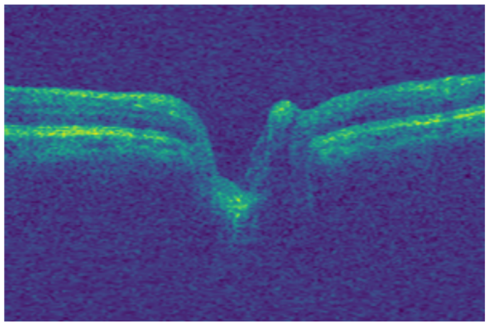

Example healthy eye from Harvard GPD


Example glaucoma eye from Harvard GDP

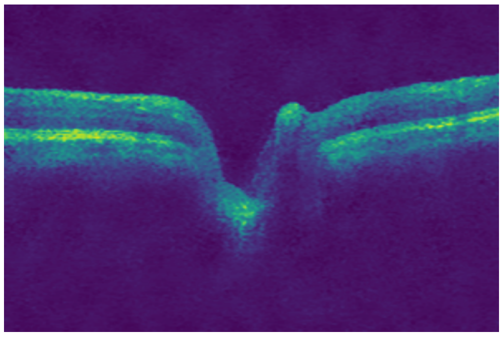

Demonstration of same scan variance, this image is from the same eye as the image directly above it

B. *Pre-Processing*

Some pre-processing was applied in this project. I wanted to apply much more pre-processing and data augmentations to the images to assist the deep networks. However, I was running into issues with exceeding memory usage (85 GB) and was limited in that aspect. Not only was I limited with the RAM usage, but I was also limited with time, and pre-processing 200k images take a long time. Because of this I wasn't able to apply as much pre-processing as I wanted to. The pre-processing that was applied was a bilateral filter applied on each image. I decided to opt for the bilateral filter because it helps deal with the significant amount of salt and pepper noise that is present in these scans. Here is an example of a before and after of the bilateral filter being applied:



Before pre-processing



After applying bilateral filter

C. *Deep Learning*

After pre-processing the images, it is time to feed them to the deep learning infrastructures. I decided to opt for the testing of 5 different deep learning algorithms to benchmark them against this dataset to see if they can be utilized to solve the problem of glaucoma classification on the OCT images. The following algorithms that were implemented:

1. VGG16 [3]
2. U-Net with naïve head [2]
3. ResNet-50 [4]
4. U-net with ResNet50 head
5. EfficientNetB0 [5]

It was also necessary to change the input from a 1 channel input to be 3 channels so that these models could digest the data. Here is the transformation that was applied:

```
single = Input(shape=(200, 300, 1))
x = Conv2D(3, (1, 1), padding='same')(single)
```

Then to interpret the output of each deep net the following pooling and activation operation was applied:

```
gap = GlobalAveragePooling2D()(output)
final = Dense(1, activation='sigmoid')(gap)
```

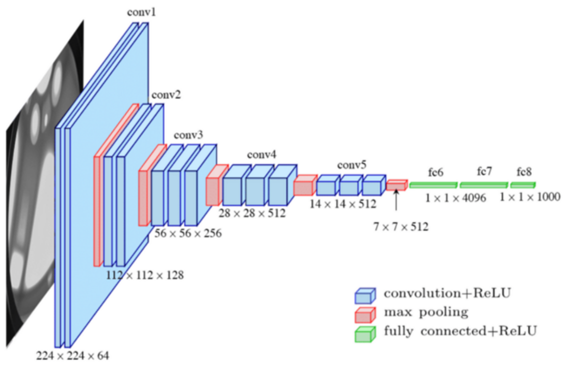
This now ensures that the deep nets can be trained for binary glaucoma classification over the OCT images.

III. Approach: Algorithms and Module Implementation Details

Before we dive into the deep learning infrastructures that were utilized on this project, let us discuss the method for training each of these algorithms. 42% of the data was used for training with 18% for validation and the last 40% for testing. The reason I didn't utilize more of the data for training was due to the RAM limitations that I had. For the deep learning models that could be downloaded from the web (VGG16, ResNet50, EfficientNetB0), I initialized the weights with the weights obtaining from imagenet. For the U-net the weights were initialized randomly. I left all of the weights in each networks to be trainable. Then Adam optimizer was utilized to train the network with a learning rate of 0.001. Binary cross entropy loss was the loss function, and a batch size of 32 over 10 epochs was utilized. The physical hardware that was utilized was a Google Colab cluster with the A100 GPU with 40 GB of GPU RAM and 85 GB of system RAM.

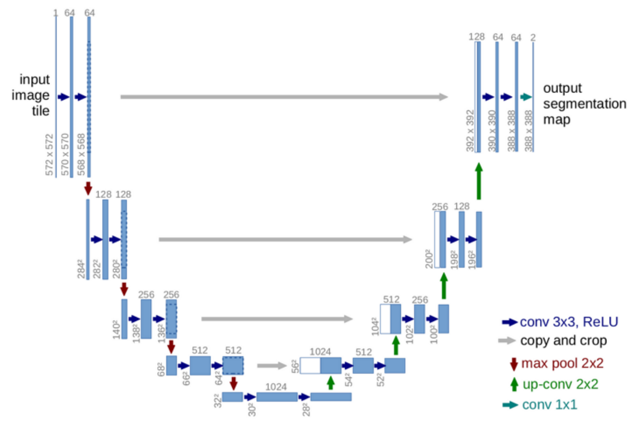
1. VGG16

The main differences between these networks that I am most interested in are the sheer size of the networks, aka the number of trainable parameters. I will provide a diagram for each network and how many training parameters they required when I trained them. VGG16 clocked in at 14,715,207 trainable parameters. I thought it required 140 million trainable parameters and was surprised that when I imported it, it was much smaller than I thought. Here is the diagram:



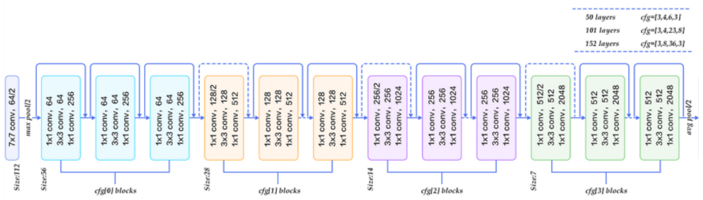
2. U-net with naïve head

The U-net with the naïve head (just the global average pooling) had the least amount of parameters in this experiment. It had 1,940,817 trainable parameters. The reason that I wanted to include the U-net in this project was due to the difference of the infrastructure as it is more of an autoencoder. I wanted to know if its unique setup would give an advantage for glaucoma classification. Here is the diagram:



3. ResNet50

The ResNet50 has 23,536,647 trainable parameters, like the U-net there are a lot of residual connections in this network. I also wanted to use this one because it is a larger network and I wanted to know if that would give an advantage. Here is the diagram:

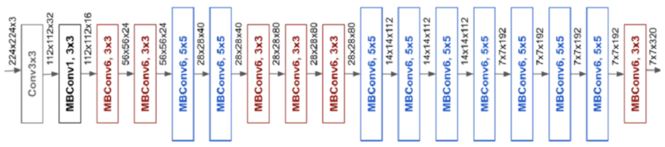


4. U-net with ResNet50 head

This is just the U-Net except attaching a ResNet50 architecture at the end. I wanted to know if this would give an advantage by having both an autoencoder and traditional CNN.

5. EfficientNetB0

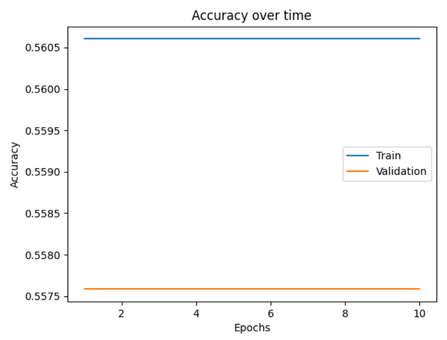
This was the final network that was tested and clocked in at 4,008,835 trainable parameters. I decided to test this model out due to it being on the smaller side compared to the other large scale CNNs and still having reasonable results. Here is the diagram:



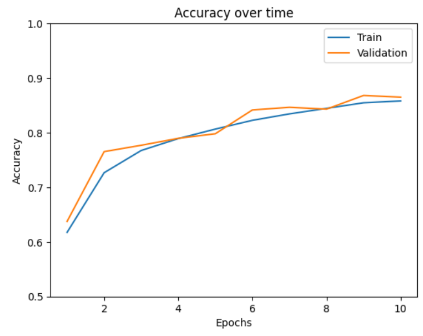
IV. Experimental Results and Discussion

Now that we have outlined the deep networks that will be tested and the method for training them it is time to train these networks. I will only provide an accuracy chart for during training since I tested so many networks. After training I will select the most promising epoch based on the validation metrics and run the models on the testing data and report the resulting table with performance metrics. Training run plots:

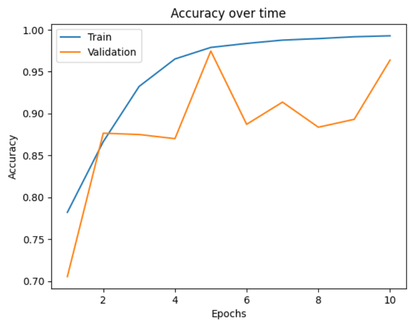
1. VGG16



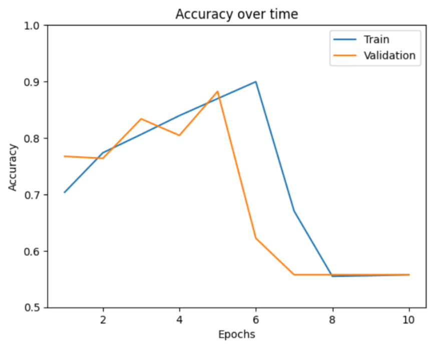
2. U-net with naïve head



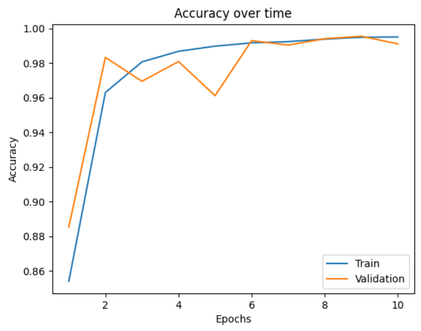
3. ResNet50



4. U-net with ResNet50 head



5. EfficientNetB0



Based on the plots I will pick the following models for further testing. VGG16, epoch 1. It didn't matter which VGG16 model I picked because at the end of the day this model did not learn anything, instead it just always selects the "non-glaucoma" class. U-net with naïve head epoch 9 will be selected due to its performance on the validation set. The ResNet50's 5th epoch will be selected. The U-net + ResNet50 epoch 5 will also be selected. Lastly, the EfficientNetB0 epoch 9 will be selected. Now it is time to test these models on the test dataset to see how they perform.

Models	Accuracy	Precision	Recall	F1
VGG16	55.25%	0%	0%	0%
U-Net Base	72.83%	71.20%	65.95%	68.48%
U-Net + ResNet50	74.82%	73.93%	67.54%	70.60%
ResNet50	72.09%	71.94%	61.67%	66.40%
EfficientNetB0	81.48%	80.06%	76.86%	78.43%

As expected, the VGG16 model did not perform that well due to it not learning anything. The U-net base, ResNet50, and U-net + ResNet50 all performed quite similarly and did not do that well overall. The EfficientNetB0 had the best performance over all of the models, but still left something to be desired when you are doing classification of whether or not a patient has a disease.

V. Conclusion and Future Work

My goal that I set out to accomplish was the classification of glaucoma over OCT images. Unfortunately, most of the models did not have desirable performance, but this also goes to show how difficult this problem is. In the future if I were to try and tackle this problem and get better results, the first thing I would do is make sure that I have a larger machine. I could also utilize online batch loading so that I don't need to load all of my data into memory at once. I could also explore utilizing classical approaches such as super pixel, since there is lots of granularities in the image which could've been confusing the networks, super pixel would capture the shape of the retina better.

References

[1] S. Jiang, Y. Peng, Y. Zhu, B. Caicedo, N. Rabinowitz, J. Kong, and L. Peng, "Harvard Glaucoma Detection and Progression: A Multimodal Multitask Dataset and Generalization-Reinforced Semi-Supervised Learning," in IEEE Transactions on Medical Imaging, vol. 39, no. 4, pp. 1121-1131, Apr. 2020.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234-241.

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

[5] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 610-619.