OCT Retinal disease image classification on mobile platforms

University of California Riverside

Professional Project Design

Faddy Sunna

MSOL Data Science
December 15, 2019

Abstract

Transforming global healthcare with Artificial Intelligence is dependent on being able to extract meaningful knowledge and insights from unstructured, unlabeled, and high dimensional biomedical data. Which can take the forms of two and three dimensional images (X-ray, MRI, CT, OCT), electronic health records, and time series sensor data(EEG, EKG). The quantity and quality at which this data is generated is ever increasing. With the recent popularization of wearable sensors, biomedical data is being generated on a large scale, without the need of a medical facility or expert. The quality of the wearable devices available to the consumer is rapidly improving, and consequently the gap with medical grade devices is shrinking. This new data is creating opportunities in the ability to monitor a range of risk factors.

Part of transforming global healthcare with A.I. is making mobile, timely, efficient, and reliable analysis of biomedical data viable and available for the average consumer, and eventually 3rd world countries. Which is were mobile computation on edge devices such as mobile phones comes in. Without the need for large expensive servers or even an internet connection, powerful deep learning frameworks can be utilized to provide useful insights anywhere. Modern deep learning frameworks have already proven themselves to be able to classify disease with accuracy, precision, and recall equal to or higher than experts.

Studies and information for the reliability and useability of these deep learning frameworks on mobile devices is limited. In this paper, I use Convolution Neural Networks to classify retinal disease in three dimensional Optical Coherence

Tomography images of retinas. Then I aim to evaluate the performance of the model running on a GPU and a mobile device.

Introduction

Biomedical data (sensory, medical imaging) is being created faster than humans can interpret. This data requires an experts time and judgment to properly interpret. Artificial intelligence is the solution for the data surplus. The proven and preferred deep learning frameworks for image classification are based on Convolutional Neural Networks (CNN). CNNs are a form of supervised deep learning, modeled after the visual cortex of a cat, specifically how it perceives images using a layered architecture of neurons. The groups of neurons each detect a specific shape, and communicate with the other groups in order to form a full image & understanding. CNNs were initially developed in the 1980's, and later popularized in the late 1990's. This was in part due to one of the major contributors to CNNs, Yann LeCun, who is also currently the VP and Chief AI Scientist at Facebook. LeCun co-authored a paper which introduces CNNs and how he applied them to recognize digits on bank checks (LeCun et al., 1998). This was the most successful implementation of CNNs at the time. Though it was still extremely limited due to the fact that creating and refining the classifiers required many human-hours, and was very computationally expensive.

Convolutional Neural Networks

Developments in GPU's and deep learning algorithms have provided significant gains in the abilities of classifiers. In 2012, Stanford University started its ImageNet

Large Scale Visual Recognition challenge, based on the Imagenet dataset which contains 1.2 million high resolution training images. The winning team had 84.6% accuracy with its AlexNet framework (Krizhevsky et al, 2012), that was based on the deep architecture pioneered by Yann LeCun in 1998. This competition has become the benchmark for computer vision, and since then all the winning teams have used CNNs, that have gone deeper each competition.

The four building blocks of a CNN are the convolution layer, non-linearity(Relu) layer, pooling layer, and fully connected layer.

1. *Convolution layer* - In this layer, features are extracted from the input image. The input image is scanned by a sliding window that reads a portion of the image at time as RGB vectors for each pixel. Calculations are performed on these values to identify features such as borders or specific shapes. The depth of features identified depends on the number of filters used in the CNN. With more filters,more features can be identified, allowing for a more fully connected network.

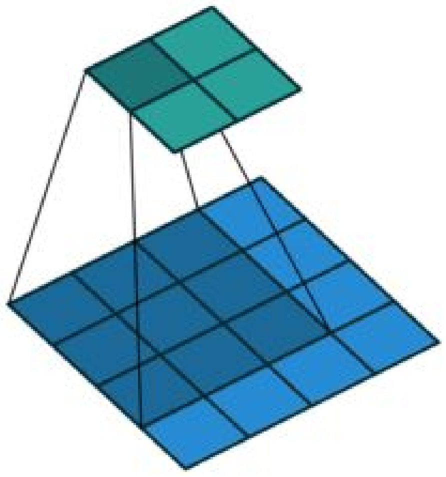


Figure 1: Example of sliding window reading portions of an image.

2. *Non-linearity layer*- This layer takes values and reduces them to a number between 0 and 1. Large negative numbers become 0, and large positive numbers

become 1, creating a piecewise activation function. The most commonly used activation function is Rectified Linear unit(ReLU).

3. *Pooling Layer* - Used to reduce the dimensionality of the feature map. Reducing noise and computation requirements. CNNs use max pooling, which creates a spatial neighborhood in the feature map, and selects the largest value from that group. Separating the most prominent features.
4. *Fully Connected layer* - The FC layer completes the CNN framework. The outputs from the Pooling and Convolutional layers represent the features of the input image. These outputs are passed into the fully connected layer, where it learns non-linear combinations of the features and performs classification of the image.

Transfer Learning

Training a successful CNN requires a large enough dataset and significant computational time/energy. When it comes to learning new skills, humans don't always need to start from scratch. We are able to transfer our knowledge from previously learned experiences onto new tasks. For example, learning to ride a skateboard will make it easier to ride a snowboard. And understanding statistics makes it easier to understand machine learning. The same concept can be utilized in deep learning with transfer learning. Transfer learning allows us to take a pre-trained model, use what it has previously learned, and fine tune it to a specific task. Reducing the need for a vast dataset and cutting computational requirements. Transfer learning is often used with

CNNs, and most models are based on the ImageNet dataset. Which contains 1.2 million training images, of 1000 different classes such as various animals, objects, cars, etc.

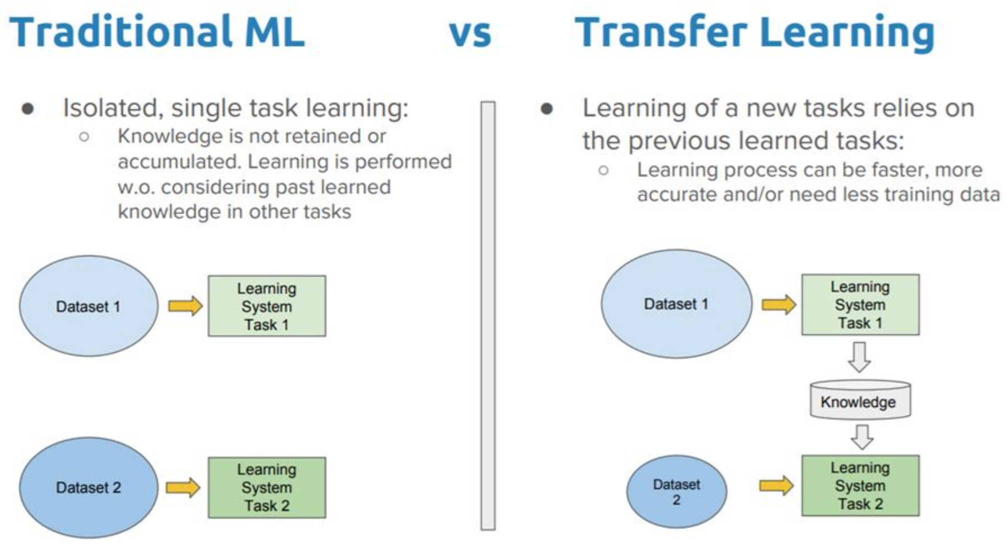

Traditional ML
vs
Transfer Learning
Isolated, single task learning:
Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
Dataset 1
Learning System Task 1
Dataset 2
Learning System Task 2
Learning of a new tasks relies on the previous learned tasks:
Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System Task 1
Knowledge
Dataset 2
Learning System Task 2

Figure 2: Tradition Ml vs Transfer learning

I will be applying transfer learning of the MobilenetV2 model to the OCT image dataset, and evaluate and compare its performance with a traditional CNN. The MobileNetV2 is a lightweight image classification model based on the ImageNet dataset, optimized for size and speed with only a small reduction of accuracy (Sandler et al., 2019). The MobileNetV2 contains nine layers, the first seven layers are frozen, and retraining is done on the final two layers for OCT image classification. This reduces its ability to recognize the previously trained ImageNet classes, but increases its ability to accurately classify the newly trained OCT images.

Project significance

In this project I will be applying CNNs to a dataset of OCT (Optical Coherence Tomography) retinal images to classify the specific form of Age Related macular Degeneration (AMD). Categorized as either Choroidal neovascularization (CNV), Diabetic Macular Edema (DME), Drusen, or Normal (No disease). OCT imaging is a non-invasive technique that uses coherance light to capture high resolution cross sections of the retina. That can later be assembled to form three dimensional images of retinal tissue. Many studies have been conducted that show that OCT continues to be an important tool in medicine. One study conducted on mice/rats shows "High-resolution spectral-domain OCT provides unprecedented high-quality 2D and 3D in vivo visualization of retinal structures "(Ruggeri et al,. 2007). Opthamologists rely heavily on OCT for the diagnosis of many eye related diseases. It is one of the most common ophthalmic diagnosis techniques with approximately 30 million scans performed each year globally (Swanson & Fujimoto, 2017). The leading cause of blindness in the United States is Age Related Macular Degeneration (AMD), and with the growing aging population, the amount of people affected by AMD is rapidly growing (Bressler et al, 2004). As shown in figure 3, there are approximately 170 million people affected by some form of AMD globally, and that number is expected to increase to 258 million by 2040, with Asia leading by a significant margin (Wong et al, 2014). The treatment can be fairly straightforward with Anti–vascular endothelial growth factor therapy (Anti-VEGF), which is already in widespread use and can treat many forms of AMD

(Kaiser et al., 2007). In some cases the drug can simply be delivered in the form of eye drops.

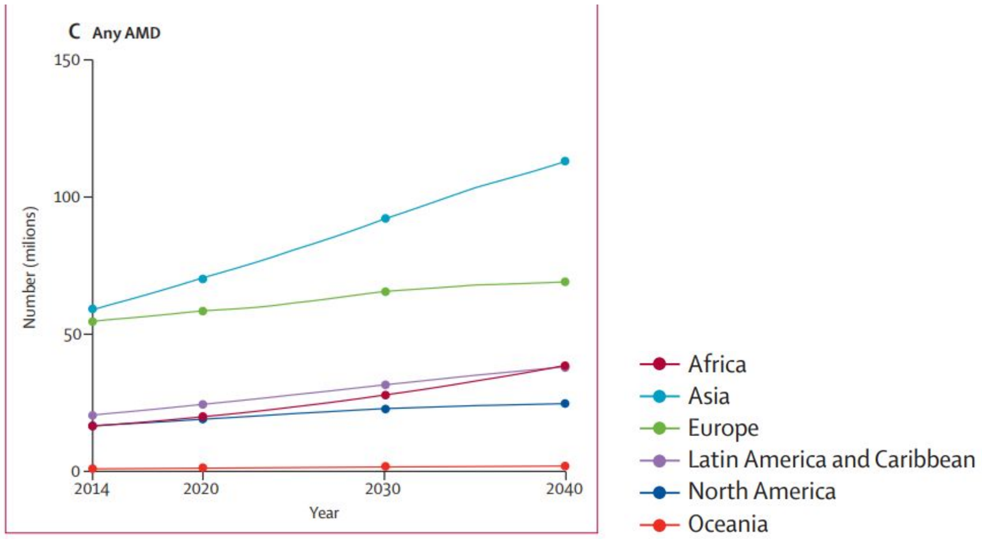


Figure 3: Expected number of people to have any form of Age related macular degeneration by 2040. (Wong et al, 2014)

CNNs running on systems with powerful GPUs have been shown to perform on par or better than experts in classifying these retinal diseases. But information is limited on the reliability, usability, and performance of such models that are optimized for and executed on mobile/edge devices. The aim of this paper is to evaluate and compare how these models perform on a mobile device, and a GPU.

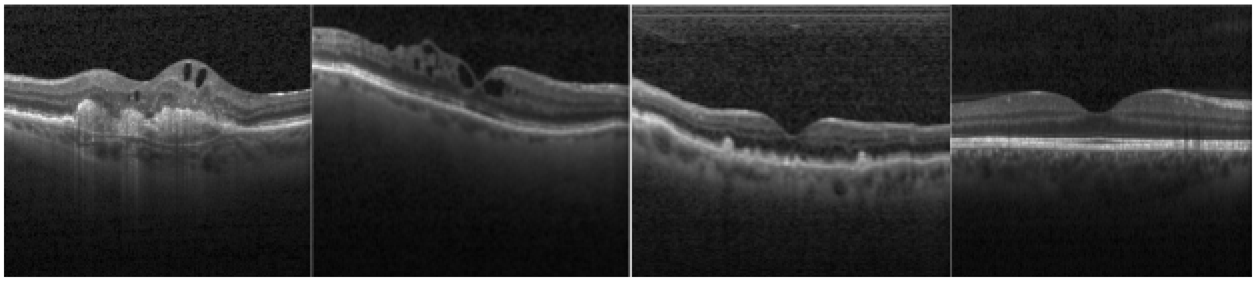


Figure 4: From left to right, OCT retinal images of CNV, DME, Drusen, and Normal.

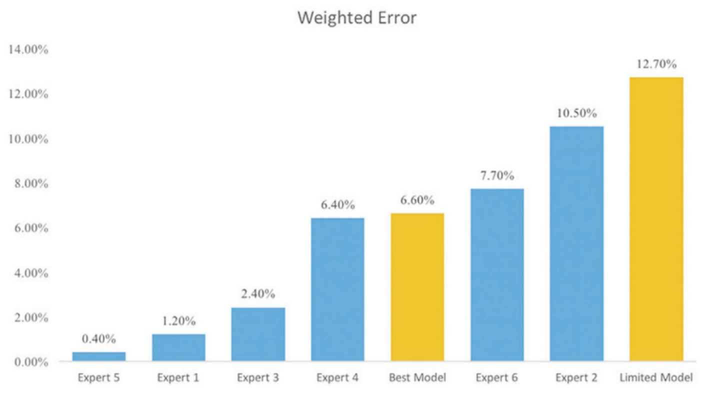
Literature review

In 2018 a combined research group implemented a CNN to classify CNV, DME, and Drusen in OCT images (Kermany et al., 2018) They used transfer learning, fine tuning the InceptionV3 model to fit their dataset of 103,802 training images. The performance of Kermany's model was evaluated by comparing the results with human experts who were given 1000 images each to classify. "Six experts with significant clinical experience in an academic ophthalmology centers were instructed to make a referral decision on each test patient using only the patient's OCT images." (Kermany et al., 2018) Their results are shown below in figures 5 and 6. Figure 5 shows that the best model ranked 5/7 among the experts, with a weighted error of 5.5%. The error is weighted in the following way:

- CNV and DME are likely to lead to blindness if left untreated. Drusen takes time to progress and is not as serious.
- Misdiagnosing someone with Drusen or Normal retinas, as CNV or DME may cause them undue distress

- But misdiagnosing someone with CNV or DME, as Drusen or Normal, may result in them not receiving the correct treatment, and ultimately blindness.

CNV and DME are grouped and labeled as “Urgent Referrals”, the weighted error is calculated by penalizing misclassifications of urgent referrals more than non urgent. They were also able to generalize their model by applying it to distinguish between bacterial and viral pneumonia in chest x-rays. The performance results were similar to those of the OCT images, though no expert comparison was done.



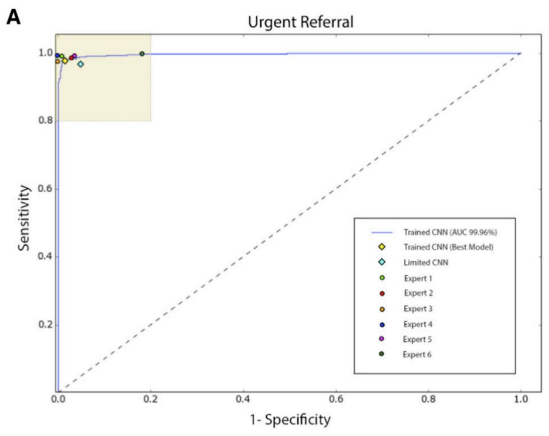


Figure 5 & 6: Error of transfer learning model vs experts in retinal disease classification, ROC curve of “Urgent referrals” (Kermany et al., 2018)

Another study completed in 2018 at the University College London in the UK created an even deeper CNN for OCT images (Fauw et al., 2018). The UK based team used the U-Net image segmentation architecture to segment the raw OCT images into 3-D tissue maps of 15 different classes such as anatomy, physiology, and image artifacts. Classification was then performed on these 3-D tissue maps. Their results were compared with a group of retina specialists, and ophthalmologists. The experts had two sessions classifying OCT images of 997 patients, one with supplemental notes, and one without. The performance comparison is shown below in figure 7. The error is weighted in a similar way to the (Kermany et al., 2018) study, but may not be exactly the same. We can see that the model scored significantly better than the experts who only had the OCT image, and comparable with the experts who had supplemental information.

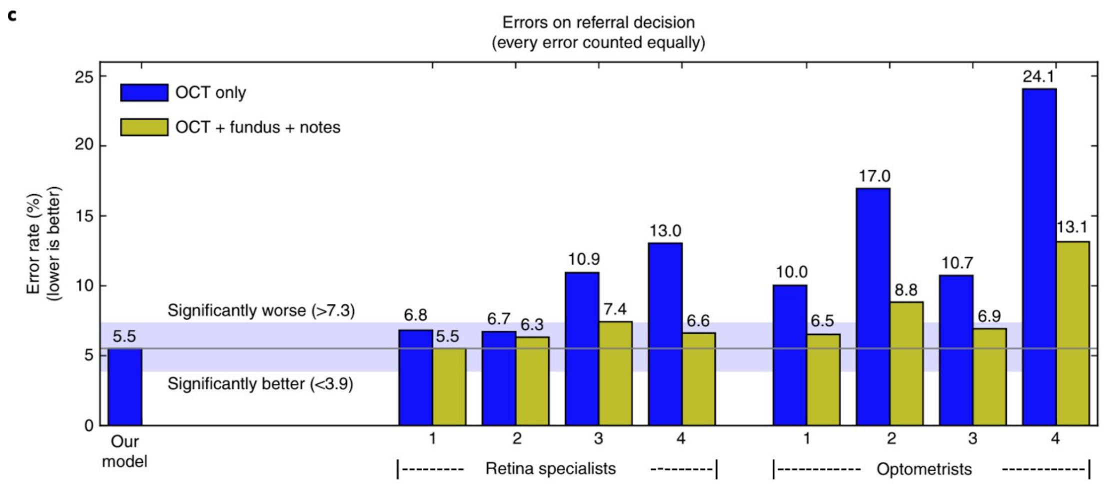


Figure 7: CNN transfer learning model vs experts conducted by (Fauw et al., 2018).

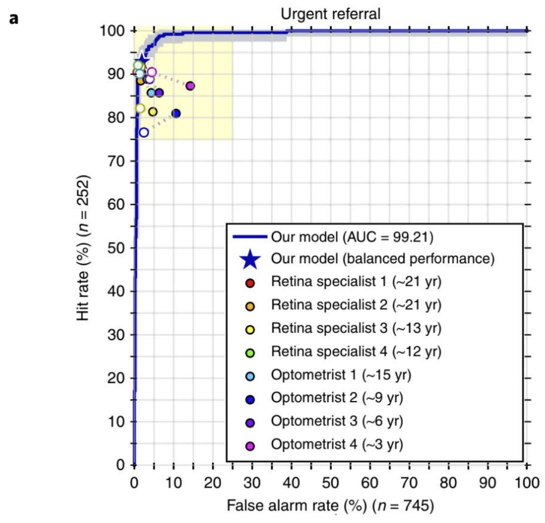


Figure 8: ROC curve of "Urgent referrals" comparing their model with the experts (Fauw et al., 2018).

(Shu Wei Ting et al., 2017) Were also able to build a deep learning system for classifying AMD from OCT images, and achieved scores comparable to professional graders.

There exists many studies that benchmark mobile phones through various deep learning tasks, but data is limited when it comes to medical deep learning and mobile devices. Judging only from the flurry of wearable sensor data being generated, the significance of mobile computation should not be underestimated. “Training on mobile devices is essential to take the collaboration between human and AI to a higher level” (Ignatov et al., 2018).

In 2018, the Department of Biomedical Engineering at Duke University developed a portable and low cost OCT system, shown in figure 8 (Kim et al., 2018). This OCT imaging system weighed only 6lb and cost would $7200. Compared with the commercially available OCT systems which cost anywhere between 30k-50k, and can weight 30-50 lb. “The performance of the system has been characterized by measuring optical parameters such as the power throughput, and lateral resolution. These compared favorably to currently available commercial OCT systems” (Kim et al., 2018)

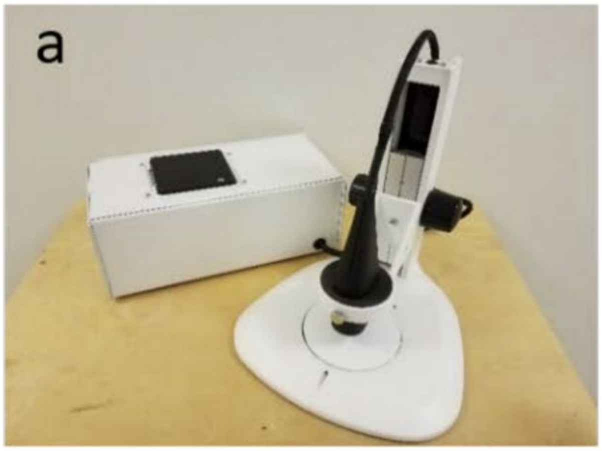


Figure 8: Lightweight portable OCT system designed by (Kim et al., 2018).

I have been able to train successful OCT classification models using transfer learning with Mobilenet, a custom VGG 16 layer CNN, and Google's Auto ML. How well do large deep learning models translate into mobile devices considering the resource limitations? Would a portable, end to end, OCT system that collects images and deliver a diagnosis be viable in the near future? These are some of the questions I aim to explore .

Methodology

Dataset: (Link and details in appendices I) The dataset used is a collections of OCT images selected from retrospective cohorts of adult patients from the Shiley Eye Institute of the University of California San Diego, the California Retinal Research Foundation, Medical Center Ophthalmology Associates, Shanghai First People's Hospital, and Beijing Tongren Eye Center between July 1, 2013 and March 1, 2017. It includes 85,236 pre-labeled OCT images with categories CNV, DME, Drusen, and Normal.

Dataset size: (Original images are 512x496 JPEG)

Training images: 75,534

Validation images: 3,136

Testing images: 5,519

Approach

Two different CNN models will be trained on the dataset of OCT images using a GPU.

- Model A: Transfer learning of MobilenetV3
- Model B: Custom 16 layer VGG CNN

Each of these models will then be evaluated and benchmarked. They will then be converted into Tensorflow Lite and Tensorflow lite quantized models to be run on a mobile device, and evaluated again.

Tensorflow model - Original Tensorflow model compiled and trained on a GPU.

Tensorflow lite - Lightweight format for mobile devices that only supports core functions and operators, with reduced input/output sizes.

Tensorflow lite quantized - Quantization further reduces the model size and memory/computational needs, usually with a minimal drop in accuracy.

Analytic methods

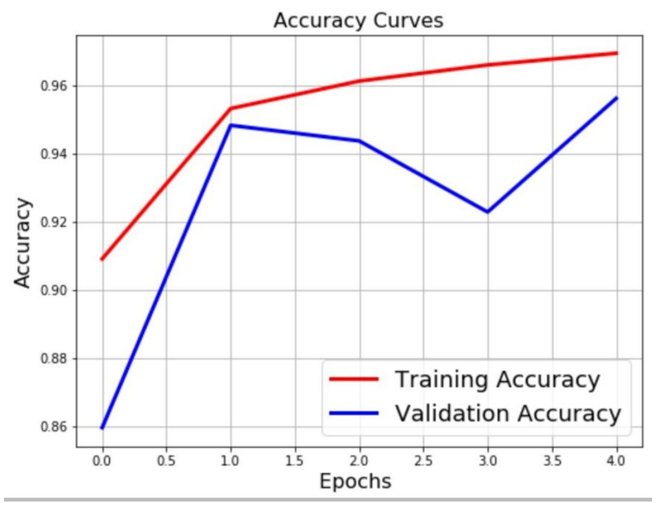
The performance of the classifiers will be evaluated with the usual multi-class metrics of precision, recall, accuracy, and specificity. The error will be weighted in a similar way to that of the performance evaluations done by (Kermany et al, 2018) and (Fauw et al., 2018). Performance on “Urgent referrals” (CNV or DME) will be evaluated via a sensitivity vs specificity ROC curve, also similarly to the previous mentioned papers. The time to perform each classification on unseen data will also be benchmarked on the GPU and mobile device.

By analyzing the performance metrics in this way, I can directly compare them with the results of previous studies which have been expert validated.

Results

The figures 9 & 10 below show the training and testing the performance of the custom 16 layer CNN. The results for the MobileNet model are in appendices section II.

Both models were successfully trained and show high accuracy after only 5 epochs. Meaning there is still plenty of room for further optimizations. The custom model scored slightly better than the transfer learning of MobileNet, with an accuracy of 96.82%, precision of 93.96%, and area under the ROC curve of 98.59%.



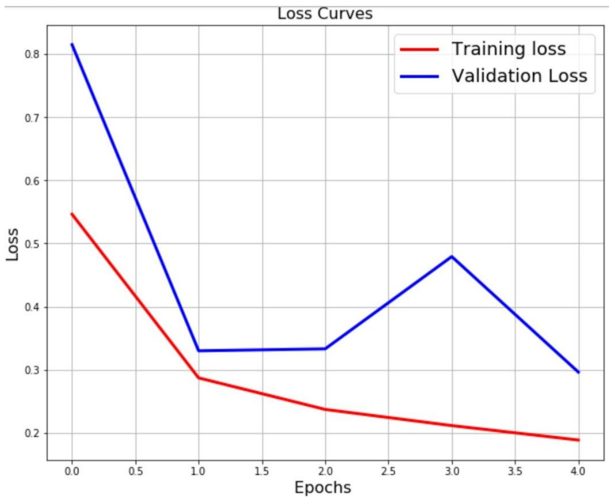
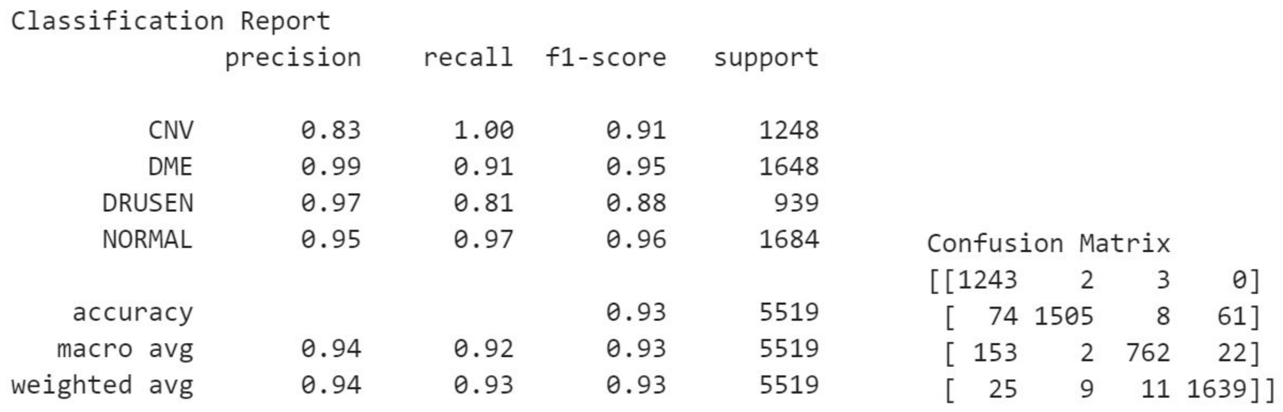


Figure 9 & 10: Custom 16 layer CNN training/validation accuracy and loss for 5 epochs

Figure 11: Performance on unseen Test images of the original Tensorflow Model A & B.

Model	Accuracy%	Precision%	Recall%	AUC%
Custom CNN	96.82	93.96	93.28	98.59
MobileNet	96.40	93.29	92.24	98.85



```
Classification Report
              precision    recall  f1-score   support

         CNV       0.83      1.00      0.91      1248
         DME       0.99      0.91      0.95      1648
      DRUSEN       0.97      0.81      0.88       939
      NORMAL       0.95      0.97      0.96      1684

    accuracy                           0.93      5519
   macro avg       0.94      0.92      0.93      5519
weighted avg       0.94      0.93      0.93      5519
```

```
Confusion Matrix
[[1243    2    3    0]
 [  74 1505    8   61]
 [ 153    2  762   22]
 [  25    9   11 1639]]
```

Figure 12: Custom CNN performance on unseen test images.

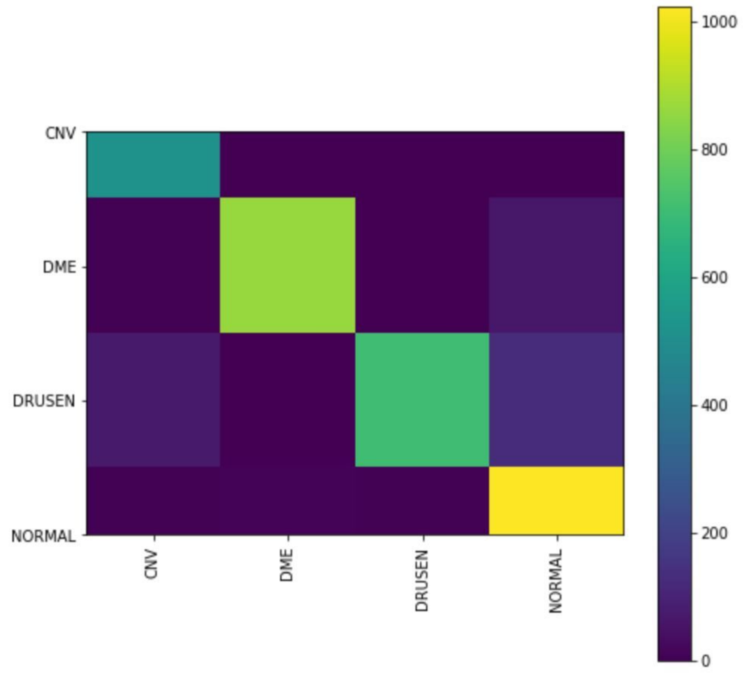
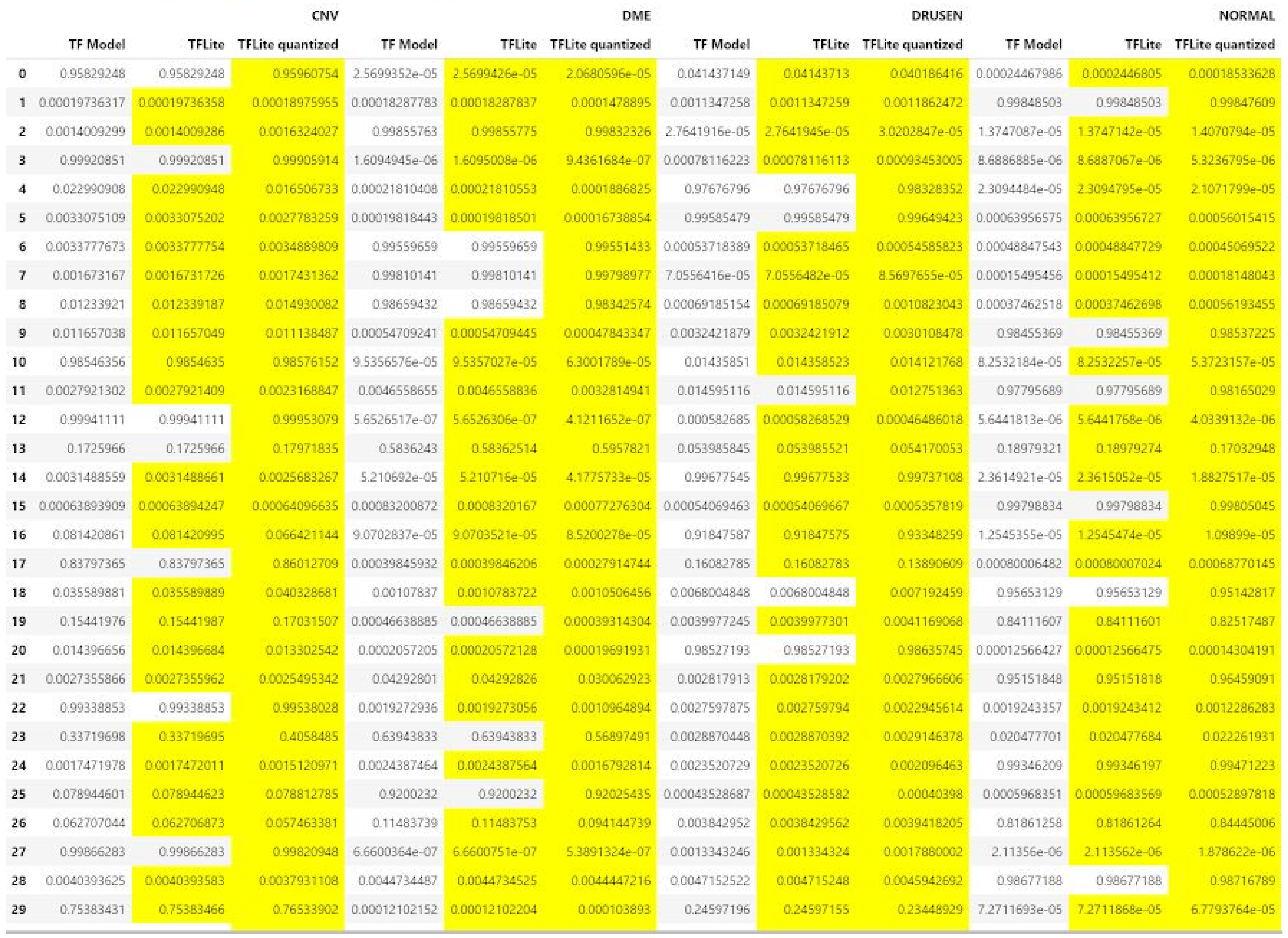


Figure 13: Custom CNN confusion matrix for predictions on unseen test images. The x-axis represents the actual label, and the y-axis is the predicted label



	CNV			DME			DRUSEN			NORMAL		
	TF Model	TFLite	TFLite quantized	TF Model	TFLite	TFLite quantized	TF Model	TFLite	TFLite quantized	TF Model	TFLite	TFLite quantized
0	0.95829248	0.95829248	0.95960754	2.5699352e-05	2.5699426e-05	2.0680596e-05	0.041437149	0.04143713	0.040186416	0.00024467986	0.0002446805	0.00018533628
1	0.00019736317	0.00019736358	0.00018975955	0.00018287783	0.00018287837	0.0001478895	0.0011347258	0.0011347259	0.0011862472	0.99848503	0.99848503	0.99847609
2	0.0014009299	0.0014009286	0.0016324027	0.99855763	0.99855775	0.99832326	2.7641916e-05	2.7641945e-05	3.0202847e-05	1.3747087e-05	1.3747142e-05	1.4070794e-05
3	0.99920851	0.99920851	0.99905914	1.6094945e-06	1.6095008e-06	9.4361684e-07	0.00078116223	0.00078116113	0.00093453005	8.6886885e-06	8.6887067e-06	5.3236795e-06
4	0.022990908	0.022990948	0.016506733	0.00021810408	0.00021810553	0.0001886825	0.97676796	0.97676796	0.98328352	2.3094484e-05	2.3094795e-05	2.1071799e-05
5	0.0033075109	0.0033075202	0.0027783259	0.00019818443	0.00019818501	0.00016738854	0.99585479	0.99585479	0.99649423	0.00063956575	0.00063956727	0.00056015415
6	0.0033777673	0.0033777754	0.0034889809	0.99559659	0.99559659	0.99551433	0.00053718389	0.00053718465	0.00054585823	0.00048847543	0.00048847729	0.00045069522
7	0.001673167	0.0016731726	0.0017431362	0.99810141	0.99810141	0.99798977	7.0556416e-05	7.0556482e-05	8.5697655e-05	0.00015495456	0.00015495412	0.00018148043
8	0.01233921	0.012339187	0.014930082	0.98659432	0.98659432	0.98342574	0.00069185154	0.00069185079	0.0010823043	0.00037462518	0.00037462698	0.00056193455
9	0.011657038	0.011657049	0.011138487	0.00054709241	0.00054709445	0.00047843347	0.0032421879	0.0032421912	0.0030108478	0.98455369	0.98455369	0.98537225
10	0.98546356	0.9854635	0.98576152	9.5356576e-05	9.5357027e-05	6.3001789e-05	0.01435851	0.014358523	0.014121768	8.2532184e-05	8.2532257e-05	5.3723157e-05
11	0.0027921302	0.0027921409	0.0023168847	0.0046558655	0.0046558836	0.0032814941	0.014595116	0.014595116	0.012751363	0.97795689	0.97795689	0.98165029
12	0.99941111	0.99941111	0.99953079	5.6526517e-07	5.6526306e-07	4.1211652e-07	0.000582685	0.00058268529	0.00046486018	5.6441813e-06	5.6441768e-06	4.0339132e-06
13	0.1725966	0.1725966	0.17971835	0.5836243	0.58362514	0.5957821	0.053985845	0.053985521	0.054170053	0.18979321	0.18979274	0.17032948
14	0.0031488559	0.0031488661	0.0025683267	5.210692e-05	5.210716e-05	4.1775733e-05	0.99677545	0.99677533	0.99737108	2.3614921e-05	2.3615052e-05	1.8827517e-05
15	0.00063893909	0.00063894247	0.00064096635	0.00083200872	0.0008320167	0.00077276304	0.00054069463	0.00054069667	0.0005357819	0.99798834	0.99798834	0.99805045
16	0.081420861	0.081420995	0.066421144	9.0702837e-05	9.0703521e-05	8.5200278e-05	0.91847587	0.91847575	0.93348259	1.2545355e-05	1.2545474e-05	1.09899e-05
17	0.83797365	0.83797365	0.86012709	0.00039845932	0.00039846206	0.00027914744	0.16082785	0.16082783	0.13890609	0.00080006482	0.00080007024	0.00068770145
18	0.035589881	0.035589889	0.040328681	0.00107837	0.0010783722	0.0010506456	0.0068004848	0.0068004848	0.007192459	0.95653129	0.95653129	0.95142817
19	0.15441976	0.15441987	0.17031507	0.00046638885	0.00046638885	0.00039314304	0.0039977245	0.0039977301	0.0041169068	0.84111607	0.84111601	0.82517487
20	0.014396656	0.014396684	0.013302542	0.0002057205	0.00020572128	0.00019691931	0.98527193	0.98527193	0.98635745	0.00012566427	0.00012566475	0.00014304191
21	0.0027355866	0.0027355962	0.0025495342	0.04292801	0.04292826	0.030062923	0.002817913	0.0028179202	0.0027966606	0.95151848	0.95151818	0.96459091
22	0.99338853	0.99338853	0.99538028	0.0019272936	0.0019273056	0.0010964894	0.0027597875	0.002759794	0.0022945614	0.0019243357	0.0019243412	0.0012286283
23	0.33719698	0.33719695	0.4058485	0.63943833	0.63943833	0.56897491	0.0028870448	0.0028870392	0.0029146378	0.020477701	0.020477684	0.022261931
24	0.0017471978	0.0017472011	0.0015120971	0.0024387464	0.0024387564	0.0016792814	0.0023520729	0.0023520726	0.002096463	0.99346209	0.99346197	0.99471223
25	0.078944601	0.078944623	0.078812785	0.9200232	0.9200232	0.92025435	0.00043528687	0.00043528582	0.00040398	0.0005968351	0.00059683569	0.00052897818
26	0.062707044	0.062706873	0.057463381	0.11483739	0.11483753	0.094144739	0.003842952	0.0038429562	0.0039418205	0.81861258	0.81861264	0.84445006
27	0.99866283	0.99866283	0.99820948	6.6600364e-07	6.6600751e-07	5.3891324e-07	0.0013343246	0.001334324	0.0017880002	2.11356e-06	2.113562e-06	1.878622e-06
28	0.0040393625	0.0040393583	0.0037931108	0.0044734487	0.0044734525	0.0044447216	0.0047152522	0.004715248	0.0045942692	0.98677188	0.98677188	0.98716789
29	0.75383431	0.75383466	0.76533902	0.00012102152	0.00012102204	0.000103893	0.24597196	0.24597155	0.23448929	7.2711693e-05	7.2711868e-05	6.7793764e-05

Figure 14: Classifications confidence comparison of Tensorflow, lite, and quantized models.

Shown in figure 14 is the confidence comparison for all three model formats, on 29 test images. Each row represents a different image, and the value is the classifiers confidence for that class. The cells that contain a different confidence value from the original Tensorflow model are highlighted in yellow. We can see that the Tensorflow lite model has small variations from the original model, but these are insignificant because it will output the same result. But in about 7 occasions, we can see the quantized Tensorflow lite model make very different predictions.

The Tensorflow lite and lite quantized models were implemented into an Android application, and run on a mobile device. The time benchmark is shown below.

Model format	Average inference time (ms)
Tensorflow lite	186
Tensorflow lite quantized	110

Figure 15: Average time to classify each image, running on a mobile phone.

Both lite formats performed very quickly with 5-10 images classified per second on a mobile device.

Conclusion

I was able to train a CNN classifier that performs comparably with the models and expert testing conducted by (Kermany et al., 2018) and (Fauw et al., 2018). With an unweighted error of 3.12% after only 5 epochs, the custom 16 layer CNN is performing well even with such little training and optimization. The CNN models were then converted into Tensorflow lite and lite quantized formats. The Lite un-quantized model performed similarly to the original Tensorflow model. While the quantized model had a decreased accuracy. The difference between these two models running on a mobile device comes down to 90 milliseconds. With a medical dataset, classifier performance is an extremely high price to pay for such a small speed improvement. The TF lite format more than fast enough with an average inference time of 186 ms. The Tensorflow to lite conversion process has room for optimization. And the inference speed can be slowed down if it means increased accuracy. In comparison, some modern ultrasounds/MRI/CT imagers can take a few minutes to load the images.

The Tensorflow lite model performed very well and showed little degradation from the original model. This promising for the future of AI, because it opens the door for devices such a light and mobile OCT imaging system than can also diagnose a patient on the spot.

References

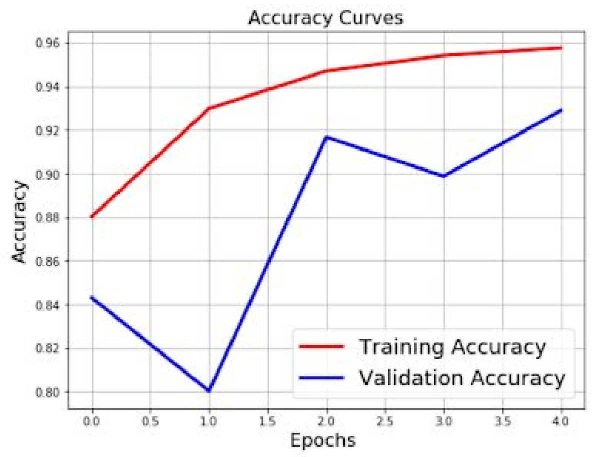
1. Ergun, Erdem, et al. “Incidence of Patients Presenting with Exudative Maculopathy and Neovascular Retinal Disease in an Urban Population.” *Wiener Klinische Wochenschrift*, vol. 116, no. 21-22, 2004, pp. 737–743., doi:10.1007/s00508-004-0262-2.
2. Fauw, Jeffrey De, et al. “Clinically Applicable Deep Learning for Diagnosis and Referral in Retinal Disease.” *Nature Medicine*, vol. 24, no. 9, 2018, pp. 1342–1350., doi:10.1038/s41591-018-0107-6.
3. Ignatov, Andrey, et al. “AI Benchmark: Running Deep Neural Networks on Android Smartphones.” *Lecture Notes in Computer Science Computer Vision – ECCV 2018 Workshops*, 2019, pp. 288–314., doi:10.1007/978-3-030-11021-5_19.
4. Jaffe, Glenn J., and Joseph Caprioli. “Optical Coherence Tomography to Detect and Manage Retinal Disease and Glaucoma.” *American Journal of Ophthalmology*, vol. 137, no. 1, 2004, pp. 156–169., doi:10.1016/s0002-9394(03)00792-x.
5. Kaiser, P. K., and D. V. Do. “Ranibizumab for the Treatment of Neovascular AMD.” *International Journal of Clinical Practice*, vol. 61, no. 3, 2007, pp. 501–509., doi:10.1111/j.1742-1241.2007.01299.x.
6. Kermany, Daniel s, et al. “Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning.” *Cell*, vol. 172, no. 1122, ser. 1131, 2018. *1131*.

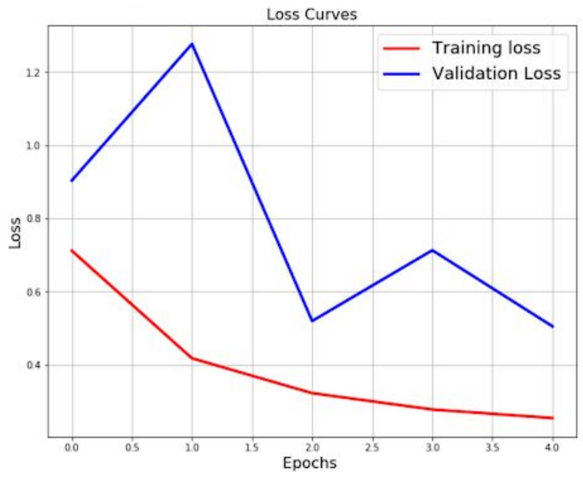
7. Kourtis, Lampros C., et al. “Digital Biomarkers for Alzheimer’s Disease: the Mobile/Wearable Devices Opportunity.” *Npj Digital Medicine*, vol. 2, no. 1, 2019, doi:10.1038/s41746-019-0084-2.
8. Krizhevsky, Alex, et al. “ImageNet Classification with Deep Convolutional Neural Networks.” *Communications of the ACM*, vol. 60, no. 6, 2017, pp. 84–90., doi:10.1145/3065386.
9. Lecun, Y., et al. “Gradient-Based Learning Applied to Document Recognition.” *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324., doi:10.1109/5.726791.
10. Lee, Susan S., and Michael R. Robinson. “Novel Drug Delivery Systems for Retinal Diseases.” *Ophthalmic Research*, vol. 41, no. 3, 2009, pp. 124–135., doi:10.1159/000209665.
11. Miotto, Riccardo, et al. “Deep Learning for Healthcare: Review, Opportunities and Challenges.” *Briefings in Bioinformatics*, vol. 19, no. 6, 2017, pp. 1236–1246., doi:10.1093/bib/bbx044.
12. Piau, Antoine, et al. “Current State of Digital Biomarker Technologies for Real-Life, Home-Based Monitoring of Cognitive Function for Mild Cognitive Impairment to Mild Alzheimer Disease and Implications for Clinical Care: Systematic Review (Preprint).” 2018, doi:10.2196/preprints.12785.
13. Song, Ge, et al. “Design and Implementation of a Low-Cost, Portable OCT System.” *Biophotonics Congress: Biomedical Optics Congress 2018 (Microscopy/Translational/Brain/OTS)*, 2018, doi:10.1364/ots.2018.of3d.7.

14. Ting, Daniel Shu, et al. “Development and Validation of a Deep Learning System for Diabetic Retinopathy and Related Eye Diseases Using Retinal Images from Multiethnic Populations with Diabetes.” *American Medical Association* , vol. 318, no. 22, ser. 2211-2223, 2017, pp. 1–13. *2211-2223*, doi:10.1001/jama.2017.18152.
15. Wong, Wan Ling, et al. “Global Prevalence of Age-Related Macular Degeneration and Disease Burden Projection for 2020 and 2040: a Systematic Review and Meta-Analysis.” *The Lancet Global Health*, vol. 2, no. 2, 2014, doi:10.1016/s2214-109x(13)70145-1.
16. Sandler, Mark, et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, doi:10.1109/cvpr.2018.00474.

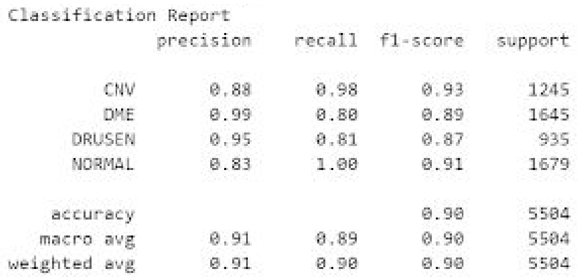
Appendix

I. Dataset
    A. https://www.kaggle.com/paultimothymooney/kermany2018
II. Hardware
    A. All the models were trained on a PC running Ubuntu, with Tensorflow and Keras in Python.
    B. GPU: Nvidia 2070 Super 8GB, CPU: AMD Ryzen 2600x 6 core
    C. Mobile Device: Samsung Note 8 (Released 2017) Snapdragon 835 processor
III. Limitations
    A. I was unable to do more direct accuracy comparison with the Tensorflow lite models due to input/output limitations of the lite models. As well as limited GPU memory, which only allowed for a image batch size of 32.
IV. Image sources
    A. Figure 1: https://goo.gl/nrMk2P
    B. Figure 2: https://cdn-images-1.medium.com/max/1440/1*9GTEzcO8KxxrfutmtsPs3Q.png
V. MobileNet performance data

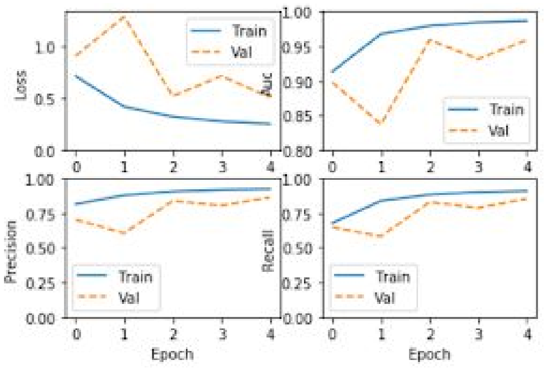




App Fig 1 & 2: (above)MobileNet training and validation accuracy/loss curves on OCT dataset

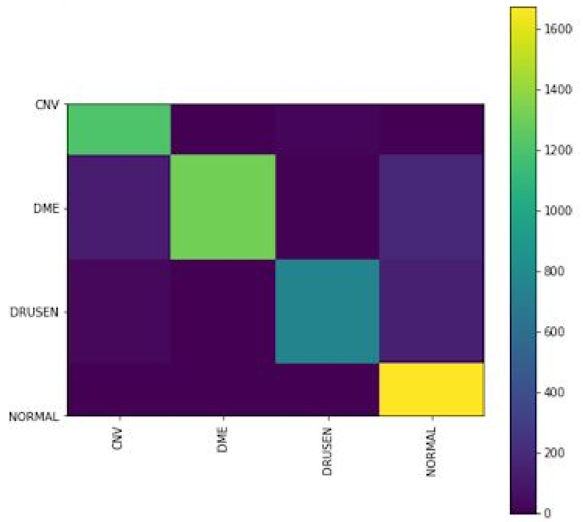


```
Classification Report
              precision    recall  f1-score   support

         CNV       0.88      0.98      0.93      1245
         DME       0.99      0.80      0.89      1645
      DRUSEN       0.95      0.81      0.87       935
      NORMAL       0.83      1.00      0.91      1679

    accuracy                           0.90      5504
   macro avg       0.91      0.89      0.90      5504
weighted avg       0.91      0.90      0.90      5504
```



App Fig 3 & 4: (above)MobileNet performance evaluation on unseen images.

App Fig 5: (below)MobileNet confusion matrix evaluation on unseen images.


CNV
DME
DRUSEN
NORMAL