Retinal Optical Coherence Tomography - Disease identification using Transfer Learning methods

K Satya Bharathi - 185001138

Sujay Sathya - 185001174

Vaibhav - 185001185

Srinath S - 185001205

UCS - 1603 - Introduction to Machine Learning

Department of Computer Science and Engineering

SSN College of Engineering

Course Instructor: Dr.S.V.Jansi Rani

Abstract

Retinal optical coherence tomography (OCT) is an imaging technique used to capture high-resolution cross sections of the retinas of living patients. Approximately 30 million OCT scans are performed each year, and the analysis and interpretation of these images takes up a significant amount of time[1]. The dataset available to tackle this problem consists of images involving 3 different forms of retinal disease namely - choroidal neovascularization (CNV) with neovascular membrane and associated subretinal fluid, diabetic macular edema (DME) with retinal-thickening-associated intraretinal fluid and multiple drusen present in early age-related macular degeneration (AMD) along with images of normal retina with preserved foveal contour and absence of any retinal fluid/edema. The images are highly intricate and need to be classified into their respective categories with great accuracy in order to be put into use in the real world. To achieve this may employ the use of advanced and modern machine learning techniques that take advantage of the spatio-temporal relationship in the images provided. Thus convolutional neural networks (CNN) are used to achieve great accuracy in the same.

Keywords: Optical coherence tomography (OCT), choroidal neovascularization (CNV), diabetic macular edema (DME), age-related macular degeneration (AMD), convolutional neural networks (CNN).

Retinal Optical Coherence Tomography - Disease identification

Retinal optical coherence tomography (OCT) is an imaging technique used to capture high-resolution cross sections of the retinas of living patients. Approximately 30 million OCT scans are performed each year, and the analysis and interpretation of these images takes up a significant amount of time[1]. This proves to be a great problem to be effectively solved using deep learning techniques to achieve great accuracy so that it can be used in the real world. To solve this problem, we use the labelled dataset provided by Kermany *et.al*[2] . The dataset consists of a total of 84,495 X-ray images consisting of 4 classes namely - CNV, DME, DRUSEN and NORMAL. *Figure 1* illustrates images belonging to the various classes.

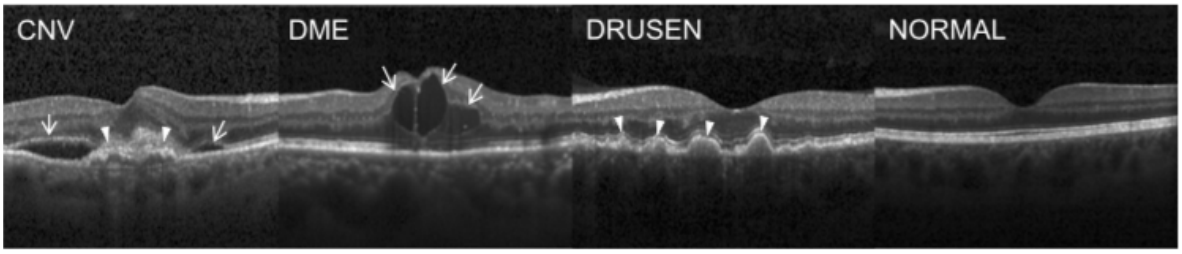

Figure 1: Shows the various classes available in the dataset

The dataset is split into train, test and validation with 83,484 images for training, 32 for validation and 968 for testing. On further analysis we were able to identify a class imbalance problem which is the presence of a few classes in higher frequency than other classes in the dataset. This issue could potentially bias the network towards predicting certain classes while training. This problem is pursued while testing and if required techniques such as SMOTE - undersampling or oversampling could potentially be employed. The same issue is shown using *Figure 2*.

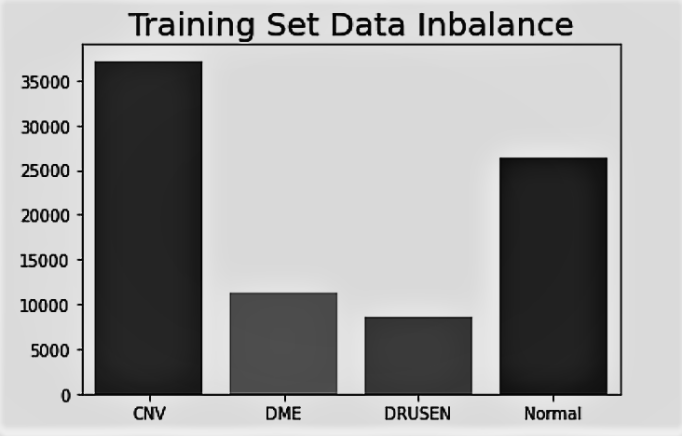

Figure 2: Displays the class imbalance problem

The solution provided by the same author[3] involved the use of convolutional networks with transfer learning which yielded them an accuracy of 96.6%. We decided to pursue this further and compare a few different transfer learning options to solve this problem.

Methodology

Used techniques

This problem requires the use of techniques that are able to capture the intricate characteristics of each image in the dataset to ensure proper medical diagnosis to be made. This is a classification problem that can be solved using convolutional neural networks (CNN).

Convolution network: Convolution as an operation can be described as the combination of two functions to produce a third. In a CNN we perform this operation on the input data over a filter/kernel to produce a feature map. The kernels are further updated continuously during backpropagation while training the neural network. Thus by minimizing the error while predicting classes - the neural network is able to learn the best kernels that produce the required feature maps required to solve the specific problem. Thus we can say that the kernel in the

convolution neural network essentially tries to form itself in the way of features present in the image. *Figure 3* shows how a typical convolution operation works.

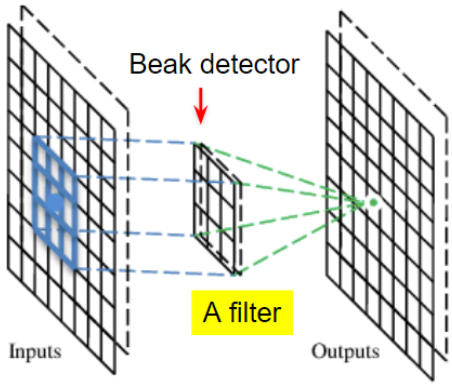

Figure 3: Convolution operation in a CNN

Transfer Learning: Transfer learning is the process of using a pre-made network architecture or the entire pretrained networks' weights without entirely training it by freezing the existing weights. Usually the final layers are removed and new layers are placed over and trained to suit the specifics of the problem at hand. Transfer learning techniques using CNNs are popular because the preliminary layers of the CNN learns rather simple features and the final layers learn the specific features. Thus by using the shallow layers as-is and by training the last layers, the neural network is able to generalize over the data present in the specific problem although the transfer learning model is trained over an entirely different network.

VGG-19 is the one of the most successful models[4] created in the recent past and it is thus used popularly in the community for transfer learning. The original network architecture as proposed by *Simoyan et.al* is given in *Figure 4*.

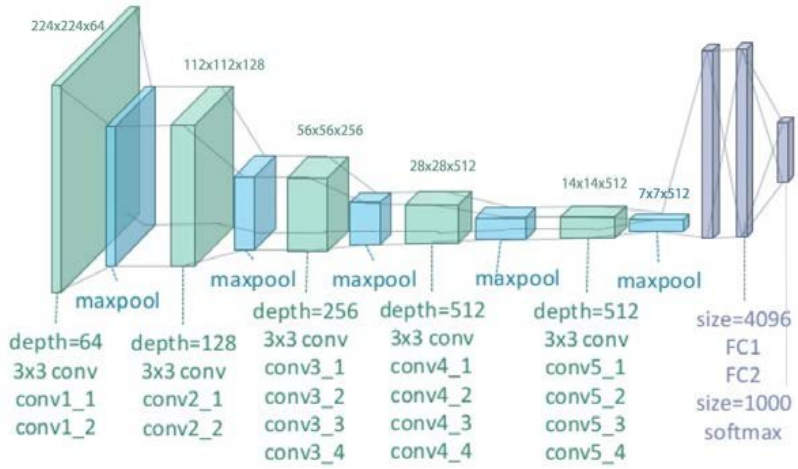

Figure 4: VGG-19 Network architecture

MobileNet-v2 is one of the more recent advancements in machine learning[5]. This network uses a lighter architecture and thus the name "mobile". The lighter network with sufficient complexity is able to be used in more day to day devices with less compute capabilities without any bottlenecks. The designers of this algorithm *Sandler et.al* proposed this algorithm to be efficient and daily complex at the same time. The original architecture is shown in *Figure 5*.

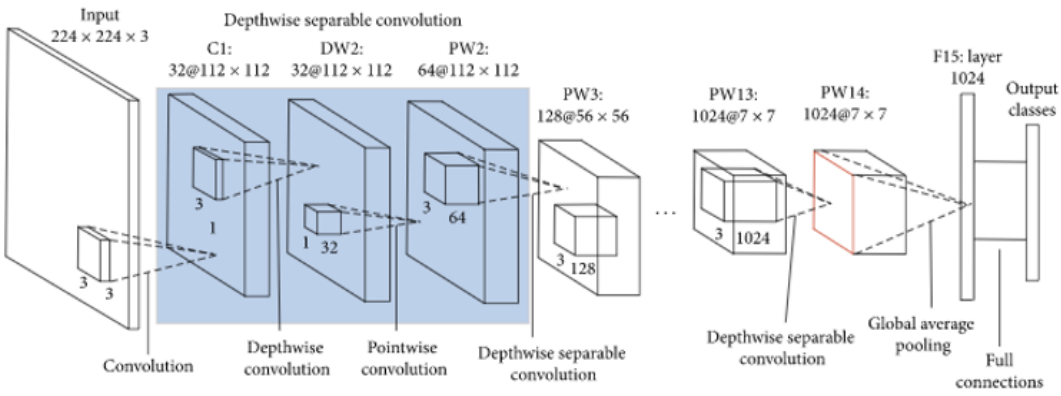

Figure 5: MobileNet-v2 Network architecture

Activation function: Activation function is key to how a neural is trained as the non linearity plays a massive role in the functions learnt by the network. When we use functions such as the rectified linear unit (ReLU) shown in *Figure 6*, the negative values are completely

diminished and essentially the backpropagation cycle is wasted as the weights aren't updated in that iteration. Thus to overcome this caveat we resort to functions such as Leak ReLU and PReLU.

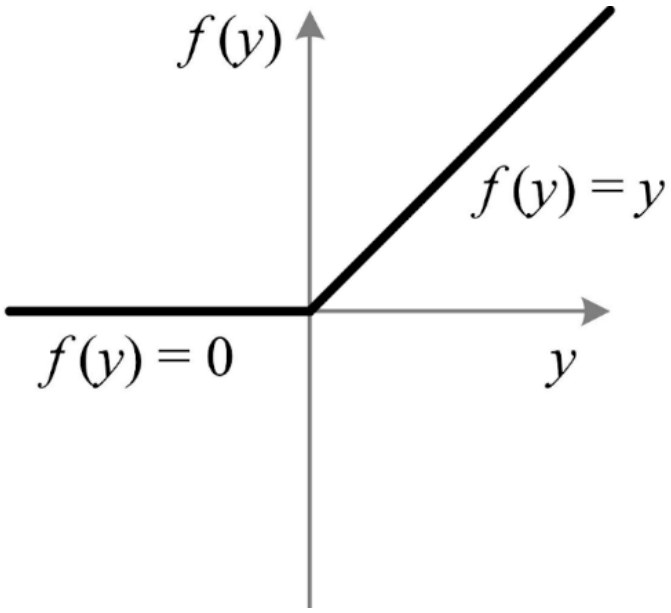

Figure 6: The ReLU activation function

Parameterized ReLU function (PReLU): The parameterized relu function is the same as ReLU on the positive side but considers a slope parameter on the negative side. This makes it better than the Leaky ReLU function which considers a static slope parameter of 0.02. The PReLU function learns this parameter dynamically through each iteration during backpropagation and is thus slightly more effective than the Leaky ReLU function. *Figure 7* shows the PReLU function with its slope parameter *a.* The functions thus helps maximizing the learning process by considering negative values as well during training and is paramount as the number of layers being trained in the transfer learning method is quite low and thus the network needs to learn the specifics of the problem within a few layers to ensure sufficient complexity.

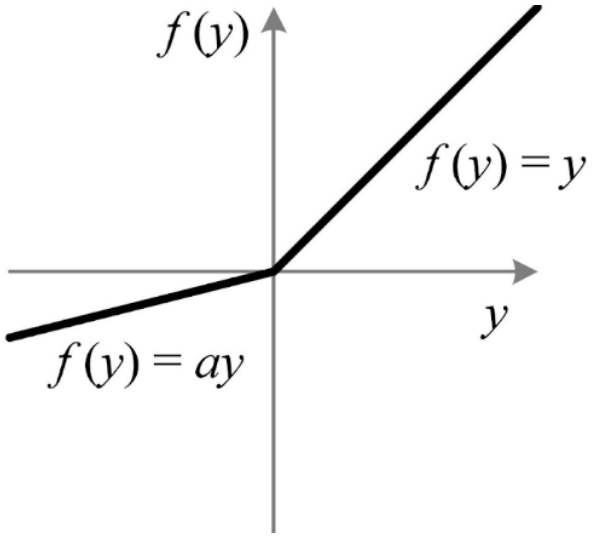

Figure 7: The PReLU activation function

Loss function - Categorical Cross-Entropy: The categorical cross entropy loss function calculates the loss of an example by computing *Equation 1*.

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Equation 1: Categorical Cross-Entropy Math

Where $\hat{\mathbf{y}}_\mathbf{i}$ is the ***i***-th scalar value in the model output, $\mathbf{y}_\mathbf{i}$ is the corresponding target value, and output size is the number of scalar values in the model output.This loss is a very good measure of how distinguishable two discrete probability distributions are from each other. In this context, $\boldsymbol{y_i}$ is the probability that event ***i*** occurs and the sum of all is $\boldsymbol{y_i}$ 1, meaning that exactly one event may occur. The minus sign ensures that the loss gets smaller when the distributions get closer to each other.

Network architecture: We decided to use 2 convolution layers with the PReLU activation function which acts as another layer after the convolution layer as the slope parameter in the PReLU needs to be learnt as well. The convolution output is fed into an intermediate dense layer after flattening and sent to the final classification layer with 4 classes. The output received from the transfer learning models is fed into the other layers and final output is received in the form of probabilities or likelihoods of image being one of the 4 classes. The entire network using *VGG-19* is displayed in *Figure 8* and *MobileNet v2* is displayed in *Figure 9*.

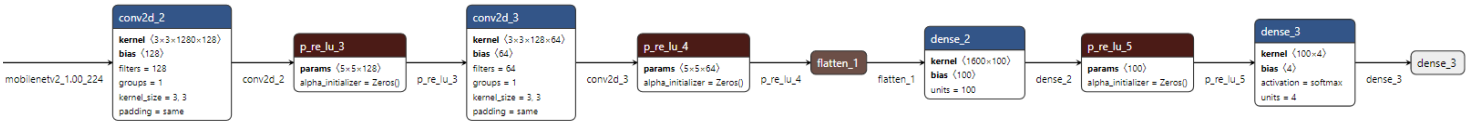

Figure 8: Network architecture using MobileNet v2

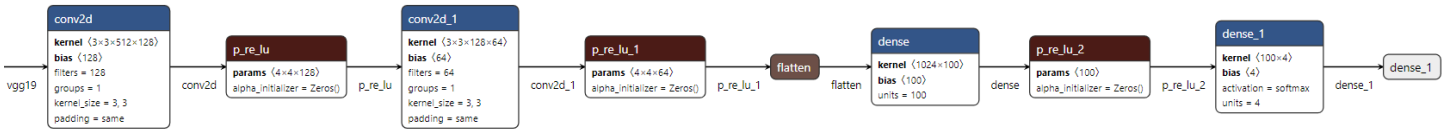

Figure 9: Network architecture using VGG 19

The weights in the transferred pretrained models are frozen and only the added layers are trained over the dataset.

Adam is used as the optimization algorithm. Adam is a culmination of years of research in optimization algorithms, specifically adaptive learning rate based algorithms. Thus as Adam is able to offer an adaptive learning rate towards the learning process, it has taken prominence in the field of machine learning in the recent past in its ability to work very well with data that is sparse as well.

Hyperparameters for both networks are given in *Table 1*.

Batch Size:	500
Epochs:	20
Learning Rate:	0.001

Table 1: Hyperparameters used in both networks during training

Assessments and Measures

The metrics we identified to test our models were accuracy, loss, precision, recall and F1 Score. We use these as classification problems such as this require the false positives and false negatives to be minimized as much as possible as it is in the medical domain. Thus to identify and optimize this we need these metrics.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

Equation 2: Formula for Recall

The metric our intuition tells us we should maximize is known in statistics as **recall**, or the ability of a model to find all the relevant cases within a dataset. The precise definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives as given in *Equation 2*. True positives are data points classified as positive by the model that actually are positive (correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect).

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Equation 3: Formula for Precision

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives as given in *Equation 3*. False positives are cases the model incorrectly labels as positive that are actually negative, or in our example, individuals the model classifies as terrorists that are not. While recall expresses the ability to find all relevant instances in a dataset, precision expresses the proportion of the data points our model says was relevant actually were relevant.

The F1 score is the harmonic mean of precision and recall taking both metrics into account in *Equation 4*.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Equation 4: Equation for F1 Score

We use the harmonic mean instead of a simple average because it punishes extreme values. A classifier with a precision of 1.0 and a recall of 0.0 has a simple average of 0.5 but an F1 score of 0. If we want to create a balanced classification model with the optimal balance of recall and precision, then we try to maximize the F1 score.

Cohen's kappa[6] measures the agreement between two raters who each classify N items into C mutually exclusive categories. The intuition for Cohen's kappa may be to think of the reliability of two raters to rate the same thing while applying a sufficient correction for the chance that they agree at random. *Equation 5* describes the formula to calculate the coefficient for Cohen's kappa. Where P_o is the probability of agreement i.e total in agreement over the total. P_e is the sum of probabilities that the raters are correct and that the raters are incorrect. These probabilities are calculated by multiplying the individual raters' probabilities of getting it correct and wrong respectively. Thus P_e gives us the probability that the raters came into agreement by chance. The importance of this statistic arises from the fact that it represents the extent to which the data collected in a given study are correct representations of the variables measured.

$$K = P_o - P_e / 1 - P_e$$

Equation 5: Cohen's Kappa coefficient formula

Results

The networks are able to achieve great accuracy over the training set of data. The metrics over the training set is described as in *Table 2* and *Table 3*.

Model:	VGG-19 based model
Accuracy:	0.9574
Loss:	0.1190
Precision:	0.9601
Recall:	0.9551
F1 Score:	0.9366
Cohen Kappa	0.9364

Table 2: Metrics achieved using VGG-19 based transfer learning over the training set

Model:	MobileNet v2 based model
Accuracy:	0.9896
Loss:	0.0311
Precision:	0.9897
Recall:	0.9893
F1 Score:	0.9847
Cohen Kappa	0.9845

Table 3: Metrics achieved using MobileNet v2 based transfer learning over the training set

The same continues over the validation set of data and the metrics for the networks are displayed in *Table 4* and *Table 5*.

Model:	VGG-19 based model
Accuracy:	0.9375
Loss:	0.2211
Precision:	0.9375
Recall:	0.9375
F1 Score:	0.9374
Cohen Kappa	0.9167

Table 4: Metrics achieved using VGG-19 based transfer learning over the validation set

Model:	MobileNet v2 based model
Accuracy:	0.9375
Loss:	0.3150
Precision:	0.9375
Recall:	0.9375

F1 Score:	0.9389
Cohen Kappa	0.9167

Table 5: Metrics achieved using MobileNet v2 based transfer learning over the validation set

Further, a confusion matrix over the training set data is shown in *Figure 10.*

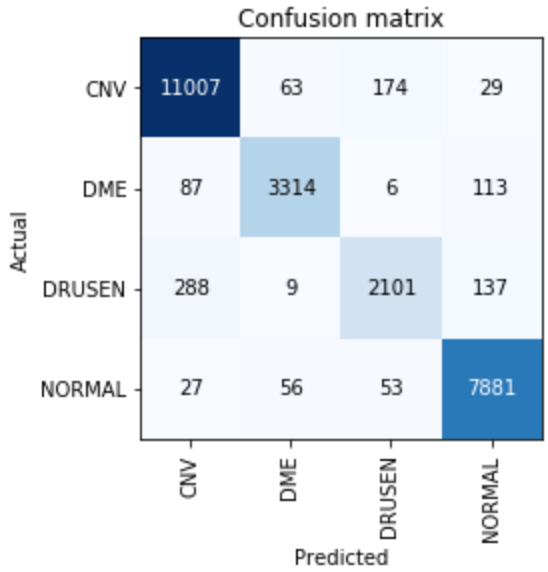

Figure 10: Confusion matrix plotted over the training set of data (using MobileNet v2)

Graphs plotting accuracy and loss with epochs over training and validation data is given in *Figures 11 & 12* for the VGG-19 based network and *Figures 13 & 14* for the MobileNet v2 based network.

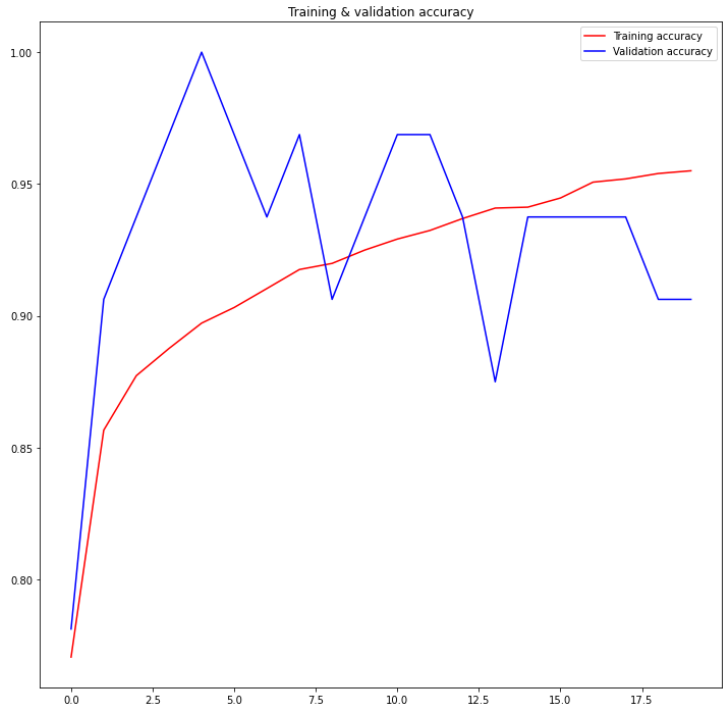

Figure 11: Training & validation accuracy for VGG-19 based network

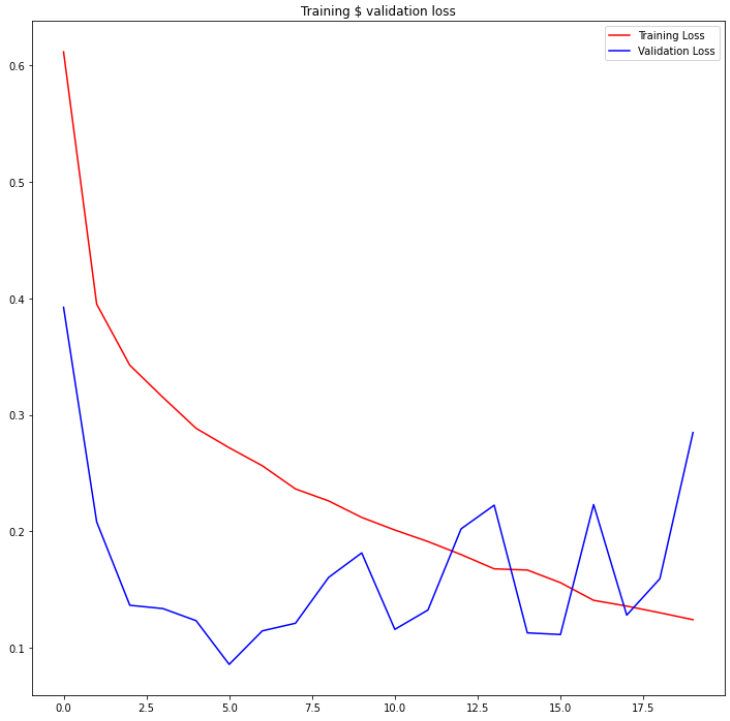

Figure 12: Training & validation loss for VGG-19 based network

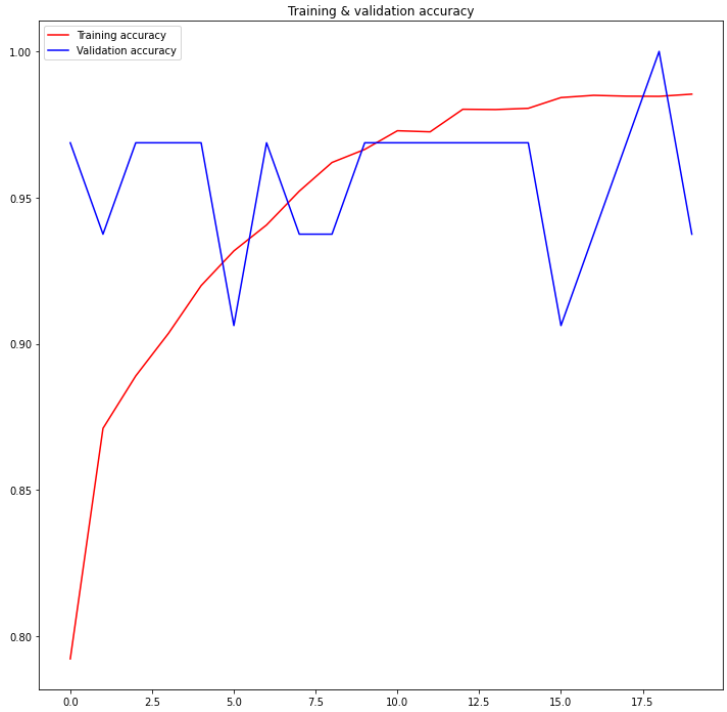

Figure 13: Training & validation accuracy for MobileNet v2 based network

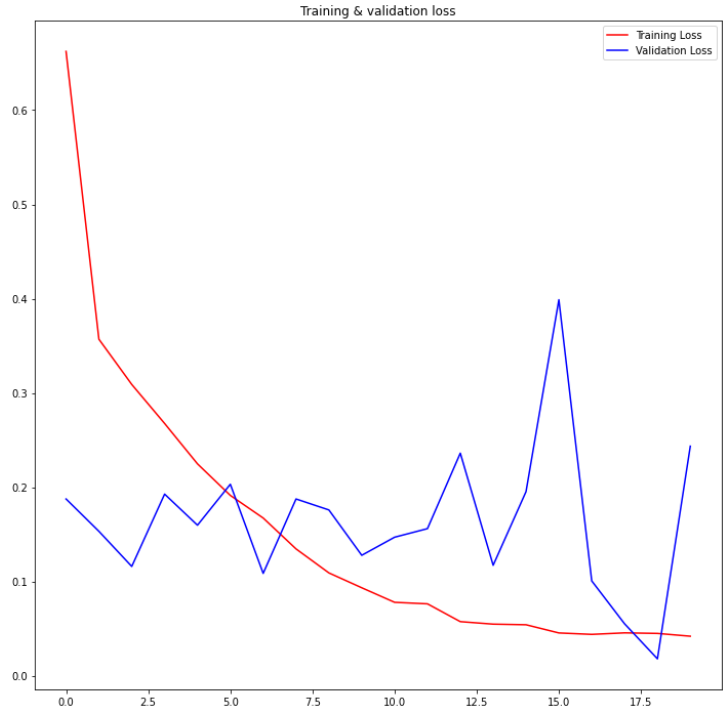

Figure 14: Training & validation loss for MobileNet v2 based network

Testing results

The models were able to perform impressively on the test set as well. The class imbalance problem does not seem to take a big toll on the capability of the model to classify the images. This may be because of the sheer amount of training data that was available, thus making it easier for the model to have enough samples to generalize over the images with minimal error. The evaluation metrics over the test data is shown in *Table 6* and *Table 7*.

Model:	VGG-19 based model
Accuracy:	0.9793
Loss:	0.0790
Precision:	0.9814
Recall:	0.9793
F1 Score:	0.9793
Cohen Kappa	0.9725

Table 6: Metrics achieved using VGG-19 based transfer learning over the test set

Model:	MobileNet v2 based model
Accuracy:	0.9700
Loss:	0.1555
Precision:	0.9700
Recall:	0.9700
F1 Score:	0.9700
Cohen Kappa	0.9601

Table 7: Metrics achieved using MobileNet v2 based transfer learning over the test set

The confusion matrices over the test set are given in *Figure 15* and *Figure 16*.

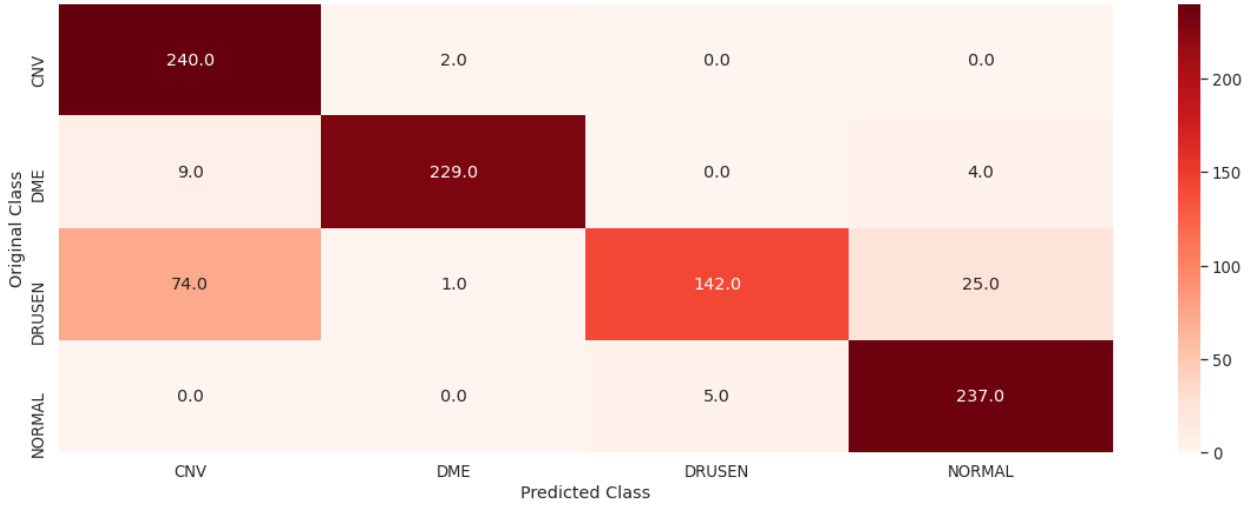

Figure 15: Confusion matrix on the test data with VGG-19 based model

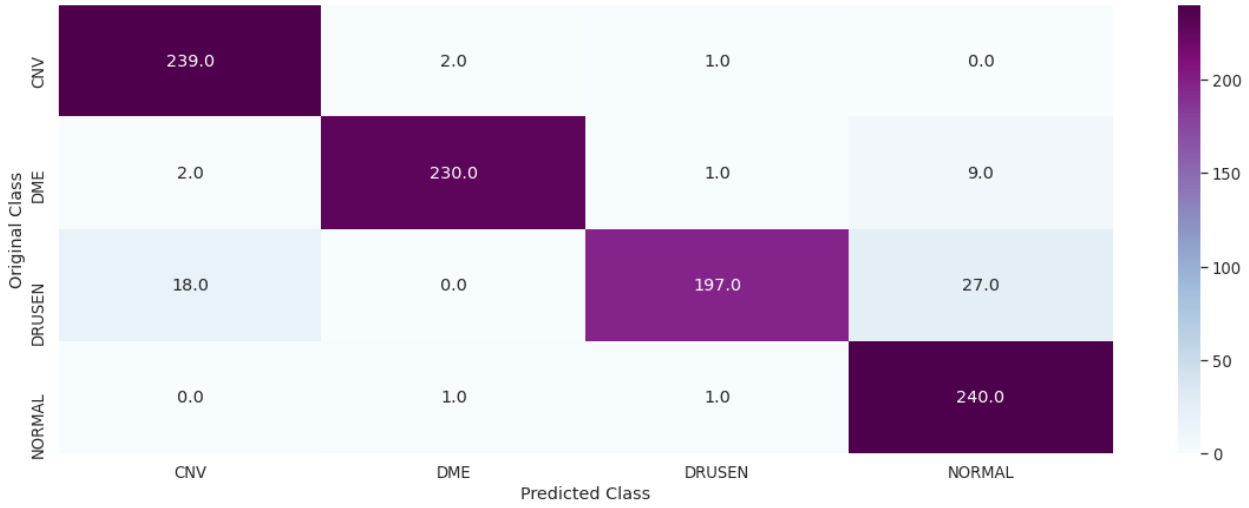

Figure 16: Confusion matrix on the test data with MobileNet v2 based model

Visualizations

To understand better how the networks are able to achieve this accuracy and learn the feature maps present in the image data, we can try to see what feature maps the convolution layers in the network have learnt. We can also see some filters selected from the network that have been used to arrive at the produced feature maps. *Figure 17* shows some feature maps that

are generated by the first convolution layer that we added over the pre trained model. *Figure 18* shows the layer above it with which we are able to see further clarity in the features identified as the depth and complexity of the network increases.

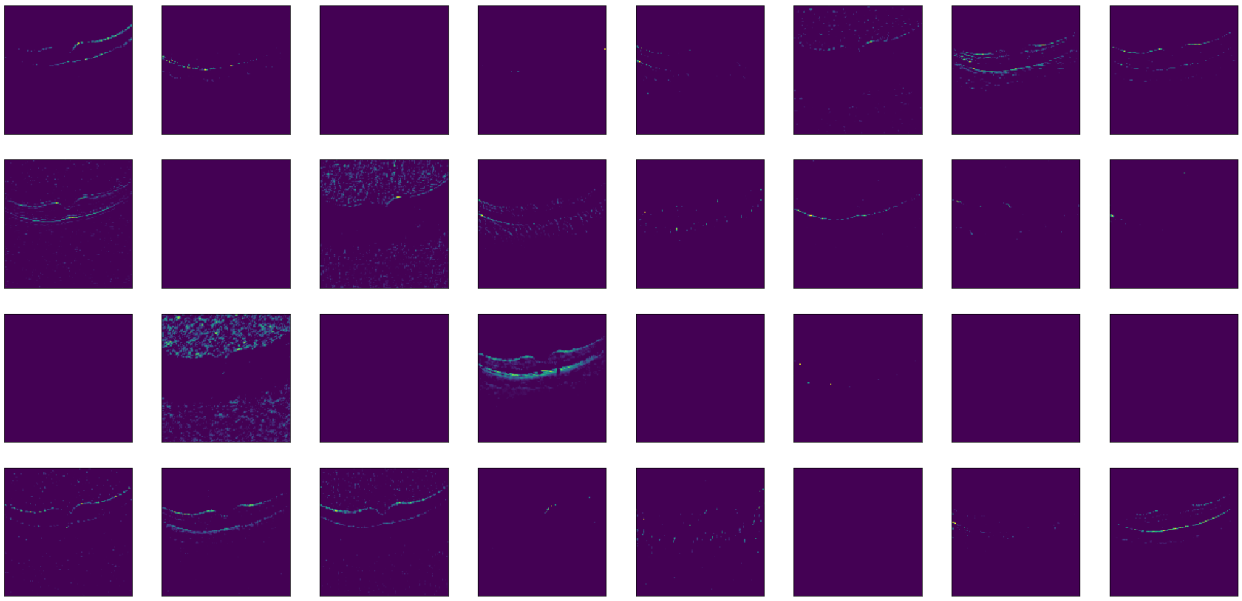

Figure 17: Examples of feature maps learnt by a convolution layer

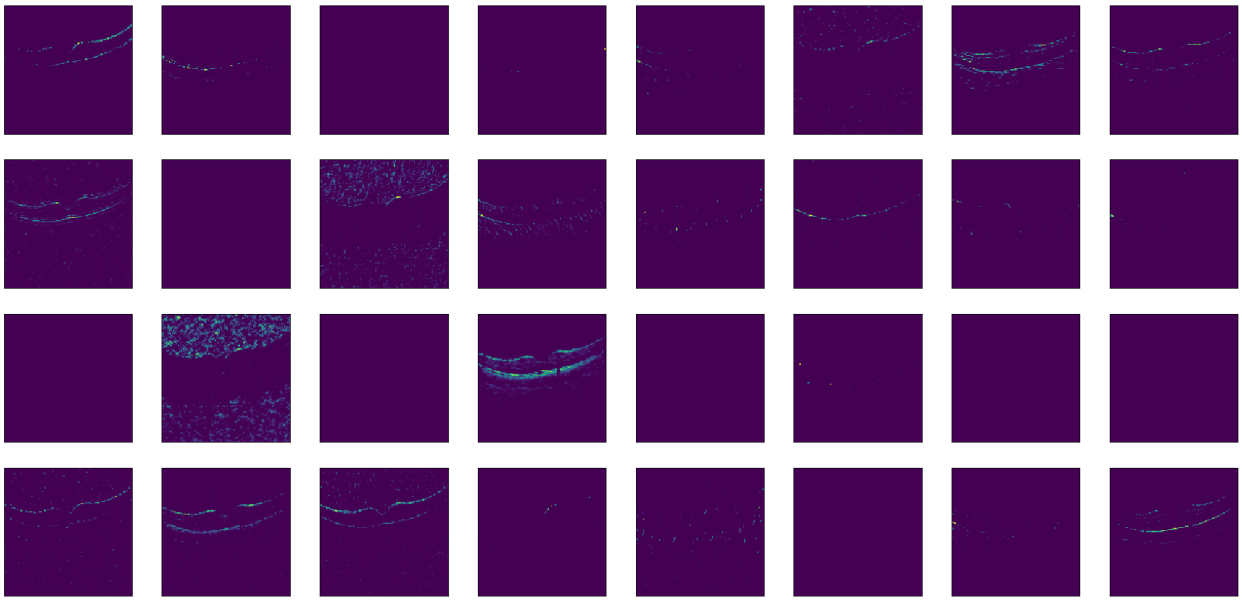

Figure 18: Examples of feature maps learnt by a deeper convolution layer

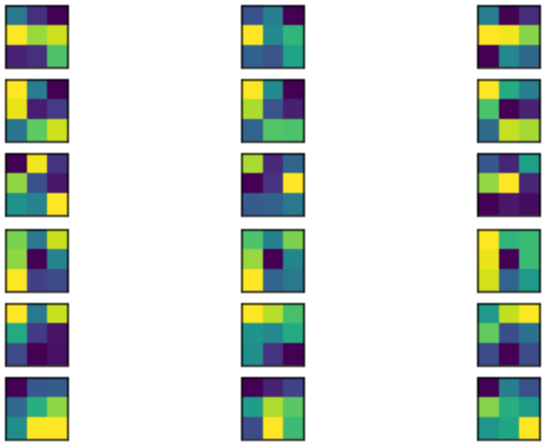

Figure 19: Examples of a few filters used in the convolution layers

Some filters selected at random are displayed in *Figure 19*. The kernels/filters along with the generated feature maps in *Figures 17 & 18* help us understand the way in which convolution layers are able to map the required spatio-temporal characteristics of an image.

Discussion & Inference

Both the networks performed well while training and were able to achieve accuracies over 95% during both training and testing. While testing however the tables turned with the VGG-19 based network performing better than that MobileNet v2 based network.

This can be attributed to the fact that the MobileNet v2 based network consists of a few bottleneck layers thereby lowering the complexity of features being learnt by the network. This is done to make the network more lightweight and so that it is able to be used in low end client hardware. The complexity of features learnt by the VGG-19 based network is more complex and the reason why MobileNet v2 based network achieves higher accuracy on the training set may be attributed to a slight possibility of overtraining, however may not be the biggest concern as the network is trained only for 20 epochs.

Observing the learnt feature maps, we are able to gather a general understanding of the type of features learnt by the network. They look very similar to the images thus leading us to believe that the networks are able to learn this problem quite accurately.

Challenges

The training of these transfer learning models take around 4.5 hours **each** accelerated by a NVIDIA P-100 class GPU. This is due to the CPU bottleneck that the large dataset poses. The *ImageDataGenerator* method in the TensorFlow Python API that is used to handle image data fetches each image separately each time while training, thus although the layers to be trained are fairly small and of low-complexity, the method struggles to fetch and keep images in the CPU while the GPU trains the model. This bottleneck means that the GPU is never actually used to full capacity because the CPU struggles to keep up with providing images while GPU performs training.

A solution we tried to solve this problem was to try and store the entire set of images in a numpy array and use that while training, however, this proved fatal with the RAM not being able to hold this capacity of data which forces the runtime to crash each time.

ImageDataGenerator fortunately has parameters that allow multiprocessing to be enabled, the no of workers be altered and allows us to set the queue size. This helped us optimize the image fetching activity according to the capability of the given runtime.

Although this provided us with a temporary solution, the training of 2 layers, ideally, shouldn't take 4 hours. Thus a method of storing images in GPU for training must be employed to perform this task efficiently.

Conclusion

Both the VGG-19 and MobileNet v2 based models are able to achieve a respectable accuracy to give a basic idea to anyone examining the X-ray. The networks are able to generalize well over the given data and perform well on the test samples as well.

The problem of Retinal OCT requires further refinement and improvement to be used in the medical industry as a standalone tool to diagnose real patients. However, this preliminary analysis can show a doctor what to look for while examining patients, thus acting as an aid to the medical professional. This can avoid wrong diagnosis or no diagnosis at all as the doctor is aware that the algorithm has found the presence of disease, making them alert to the indicators while physically examining the patient.

References

1. Swanson, E. A., & Fujimoto, J. G. (2017). The ecosystem that powered the translation of OCT from fundamental research to clinical and commercial impact [Invited]. *Biomedical optics express*, 8(3), 1638–1664. https://doi.org/10.1364/BOE.8.001638.
2. Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), "Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification", Mendeley Data, V2, doi: 10.17632/rscbjbr9sj.2
3. Daniel S. Kermany, Michael Goldbaum, Wenjia Cai, Carolina C.S. Valentim, Huiying Liang, Sally L. Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, Justin Dong, Made K. Prasadha, Jacqueline Pei, Magdalene Y.L. Ting, Jie Zhu, Christina Li, Sierra Hewett, Jason Dong, Ian Ziyar, Alexander Shi, Runze Zhang, Lianghong Zheng, Rui Hou, William Shi, Xin Fu, Yaou Duan, Viet A.N. Huu, Cindy Wen, Edward D. Zhang, Charlotte L. Zhang, Oulan Li, Xiaobo Wang, Michael A. Singer, Xiaodong Sun, Jie Xu, Ali Tafreshi, M. Anthony Lewis, Huimin Xia, Kang Zhang. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning, *Cell*, Volume 172, Issue 5, 2018, ISSN 0092-8674, https://doi.org/10.1016/j.cell.2018.02.010.
4. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
5. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

6. McHugh M. L. (2012). Interrater reliability: the kappa statistic. Biochemia medica, 22(3), 276–282.