Diabetic Retinopathy Detection using Convolutional Neural Networks, Transfer Learning and Multi-task Learning Methods

Joann Rachel Jacob
jacob.joan@northeastern.edu

Manasa Krishnan
krishnan.man@northeastern.edu

Sree Krishna Suresh
suresh.sr@northeastern.edu

Abstract

Diabetic retinopathy (DR) is one of the most threatening complications of diabetes which causes lesions on the retina that affect vision and may lead to blindness if not detected and diagnosed early. Retinal screening aids in the early detection and treatment of DR. The need for a comprehensive and automated method of DR screening has long been recognized, and previous efforts have made good progress using image classification, pattern recognition, and machine learning. The objective of this project is to develop a Deep Learning framework to improve Diabetic retinopathy screening. For this, we use Convolutional Neural Networks (CNN), Transfer Learning and Multi-task Learning Techniques to provide a solution for stage identification of diabetic retinopathy by single photography of the human fundus and ideally improving performance resulting in models with realistic clinical potential. With these algorithms/frameworks a solution is devised to find a better and optimized way of classifying the image with little pre-processing techniques. These models are then evaluated to see how they perform on this task. Additionally, multi-task learning methods are applied in efforts to infer how the model is responsive to different algorithms.

Category: Convolutional Neural Networks, Transfer Learning, Multi-task Learning

1. Introduction

Diabetic retinopathy (DR) is one of the most common and fastest-growing causes of blindness that can be avoided. It is estimated that approximately 600 million people will have diabetes by 2040, with one-third expected to have DR. During the first two decades nearly all patients with type 1 diabetes and greater than 60 percentage of patients with Type 2 diabetes will have retinopathy [16]. Research suggests that at least 90% of new cases might be avoided if the eyes were treated and monitored properly and vigilantly.

Clinical DR severity is divided into five categories which depends on the types and numbers of lesions on the retina image, as shown in Table 1. The symptoms from which we can identify that a person is suffering from DR is shown in Figure 1.

Table 1. DR stages depending on lesions classification [4]

DR Severity Level	Lesions
No Dr	No lesions
Mild DR	MA only
Moderate DR	More than just MA but less than severe DR
Severe DR	Any of the following: More than 20 intraretinal HM in each of 4 quadrants; definite venous beading in 2+ quadrants; Prominent intraretinal microvascular abnormalities in 1+ quadrant and no signs of proliferative DR
Proliferative DR	Neovascularization. Pre-retinal HM

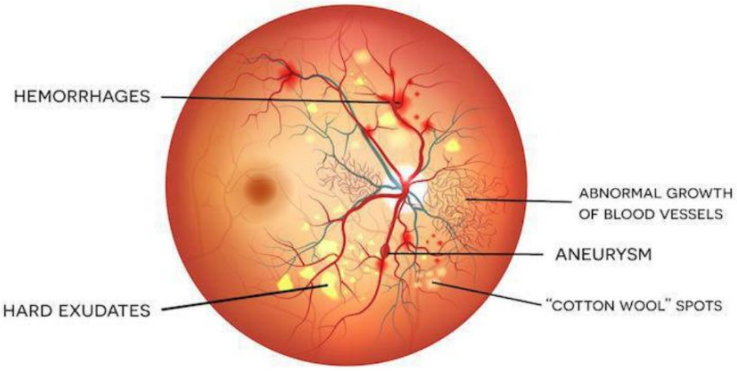

Figure 1. Diabetic Retinopathy eye image [10]

Traditional machine learning algorithms have the drawback of requiring the user to determine which attributes are meaningful in each image. Feature extraction becomes extremely difficult as the number of categories to classify grows. Here we use convolutional neural networks, transfer learning and multi-task learning to produce an efficient learning algorithm and combat issues faced by traditional algorithms.

CNN might be the best fit for image classification as it not only provides several techniques to surpass the disadvantages faced by traditional algorithms but also reduce noise in the image data and remove unwanted features by enhancing the important parts of the image. As the dataset size is relatively small, data augmentation techniques such as random rotation, horizontal flip and vertical flip are used which in turn can increase the model's accuracy by capturing various features previously unknown. Previous similar approaches in neural networks dealt with plain convolutional neural networks with a large reduction in image size. Here, we initially resized the image to 64*64 pixels to work with the models and later resized the image size to 512*512 pixels to assert whether further important features are captured by CNN and other models. As computation time for training the model with large images will be high, transfer learning is applied along with convolution and fully connected layers for modelling. Further we stacked three models under CNN architecture to form a multi-task learning method to see if the performance improves with respect to each model.

A specific limitation of this project is the storage and processing of a dataset that gets updated very frequently and that the current data set is imbalanced. There are a variety of other challenges associated with this task, as the image may be subjected to viewpoint variation, scale variation, intra-class variation, image deformation, image occlusion, illumination conditions, background clutter etc. Also, images may have artefacts, be out of focus, be underexposed or overexposed, or be taken with various devices, among other things.

2. Related Works

The initial algorithms were based on various traditional computer vision techniques and thresholds (Michael D. Abrmoff and Quellec, 2010; Christopher E.Hann, 2009; Nathan Silberman and Subramanian, 2010) [1]. Nonetheless, deep learning systems have demonstrated their dominance over conventional algorithms in classification and object detection tasks in recent years (Harry Pratt, 2016) [2]. Convolutional neural networks (CNN) have been used successfully in a variety of fields, including the detection of diabetic retinopathy (Shaohua Wan, 2018; Harry Pratt, 2016) [3]. APTOS (Asia Pacific Teleophthalmology Society) and Kaggle, a machine learning competition platform, challenged ML and DL researchers to create a five-class DR automatic diagnosis solution in 2019. (APTOS 2019 Blindness Detection Dataset) [11].

3. Data

For this project, we acquired Kaggle competition's 'The APTOS 2019 Blindness Detection' dataset [11]. It comprises of 3662 photos of human retina tagged with the five severity stages of diabetic retinopathy which was used to train the models to solve the problem. The images are of various dimensions. In this dataset 1805 of the photos are tagged as No DR, whereas the rest 1857 are tagged with other stages of DR causing the dataset to be imbalanced, the distribution for the same is shown in Figure 2. The severity is labeled from 0 to 4 with each number representing the different stages of severity namely No DR - 0, Mild -1, Moderate -2, Severe – 3, Proliferate – 4 respectively. Images of retina belonging to all classes involved are shown in Figure 3.

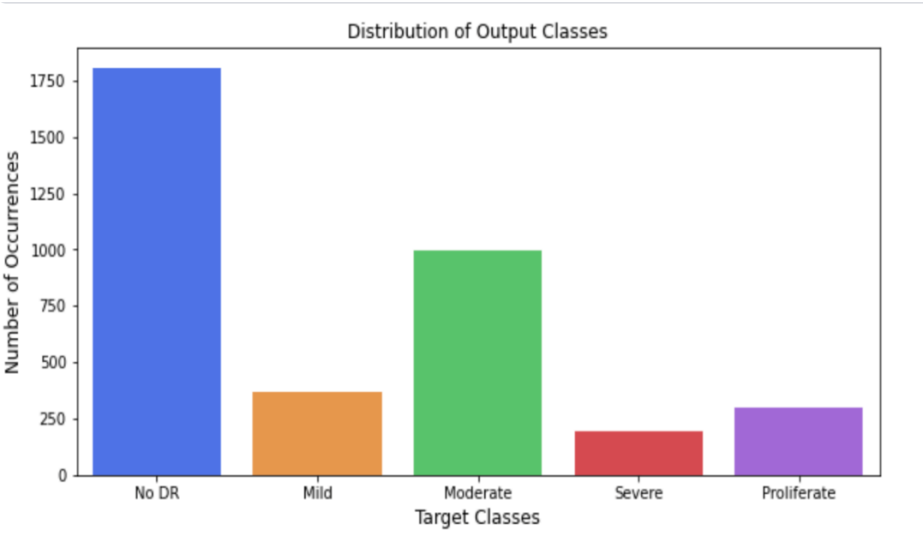

Figure 2. Distribution of Severity Classes in the dataset.

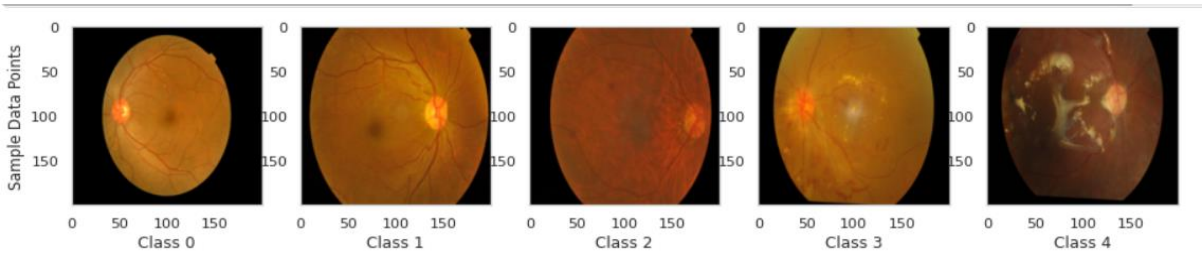

Figure 3. Various DR stages

The dataset consists of the multiple files, namely, train.csv, test.csv and folders train images and test images. Train file consists of attributes: 'id_code' and diagnosis and test file consists of attribute id_code alone. 'id_code' is the image id code which is used to map to the images in the train images and test images folder. 'diagnosis' has values that range from 0-5 and implies the various classes of DR stages.

4. Methods

4.1 Data Pre-Processing and Visualization

The train and test images we acquired for this project are coloured and of a huge size which makes it difficult and cumbersome to process the images in neural network. Thus, we convert the images to grayscale to increase our processing speed. We then use Gaussian Blurring to lessen the irregularities in the images especially around the edges and corners and noise in data. The Gaussian blur is a type of picture-blurring filter that calculates the change to implement to each pixel in the image using a Gaussian function (which also describes the normal distribution in statistics). The formula of a Gaussian function in single dimension is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution [9]. We also tried to create a circular cropping around the image centre. Figure 5 shows the images of various classes obtained after improving lighting condition using Gaussian Blur and circle cropping.

The images are flattened using T-Distributed Stochastic Neighbour Embedding (T-SNE) for visualisation purposes. T-SNE is a dimensionality reduction technique to convert images to 2 or 3 dimensions. It is a probabilistic technique that can be used when the images are of an extremely high dimension. After applying this, we can easily distinguish between each class. The first two images shown below indicate the first and final step in T-SNE differentiation of classes.

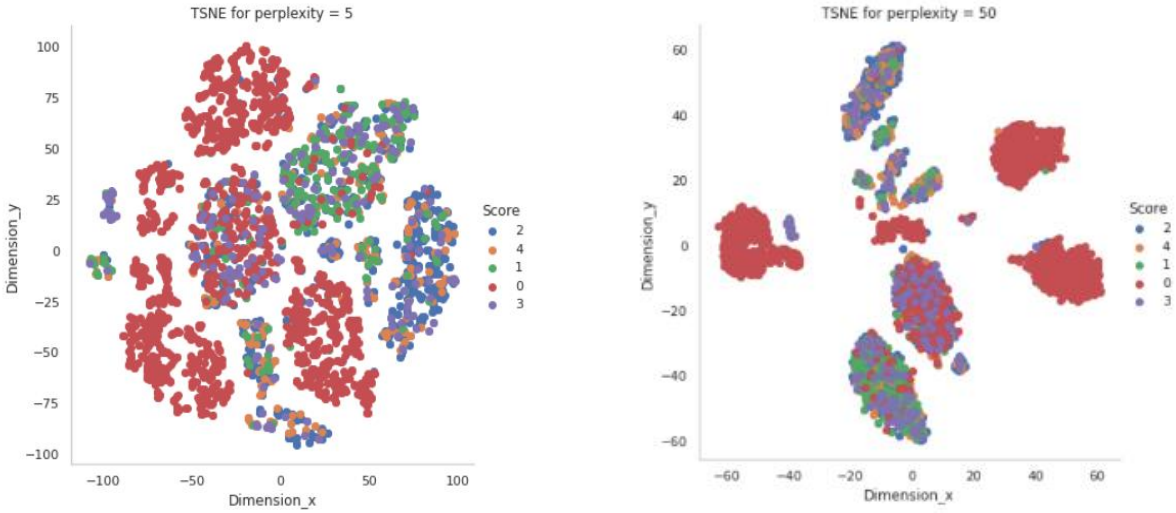

Figure 4. TSNE first and final step

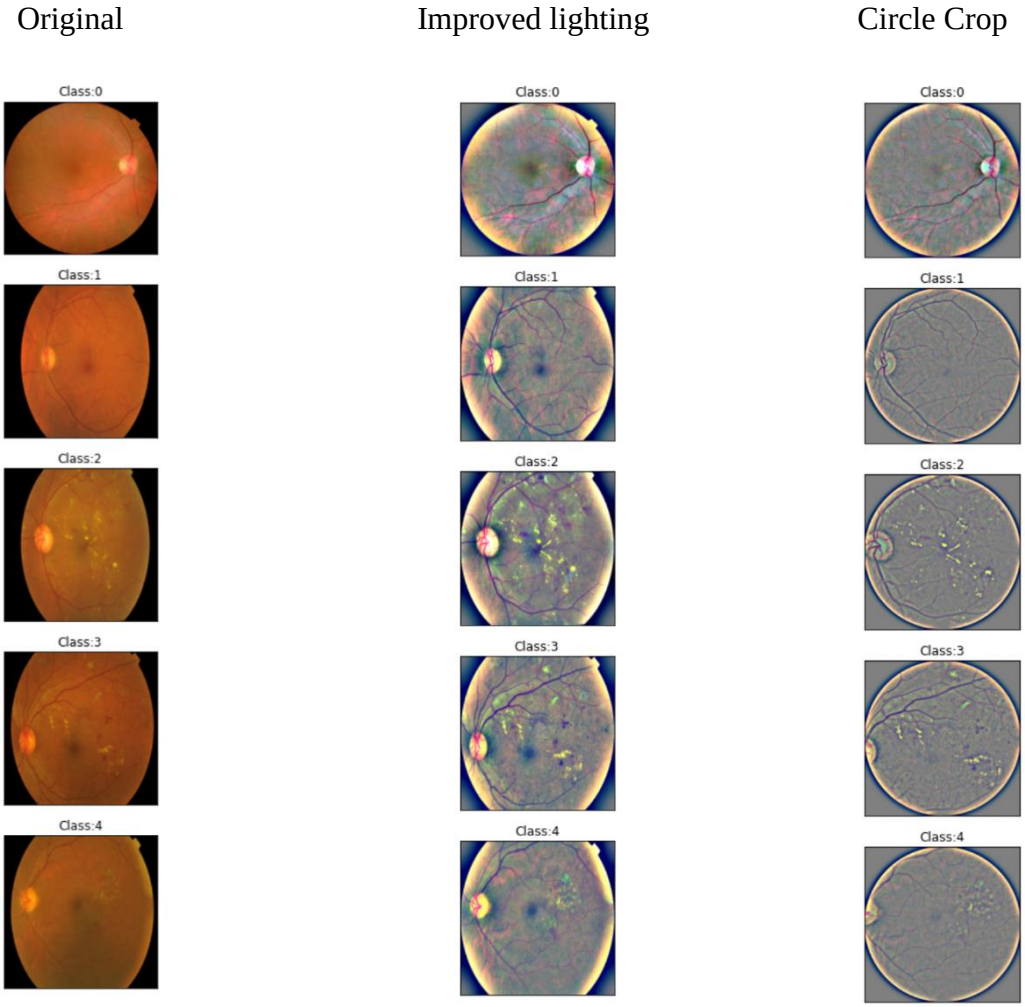

Figure 5. Images of 5 severity classes before and after pre-processing steps

4.2 Data Augmentation

Sometimes if the image is oriented in direction or any other form, the model might consider that to be the classifying factor and make decisions based on that. To prevent this, we have several forms of the same image so that we can consider every way an image can be conceived. As we handle images, brightness, frame, and other varying factors of the images may skew classification. Thus, we use data augmentation and obtain various forms of the image as in Figure 6 by rotating the image, flipping the image and so forth. When training a machine learning model, this approach acts to regularize and helps reduce overfitting. It can also be considered as a method of oversampling.

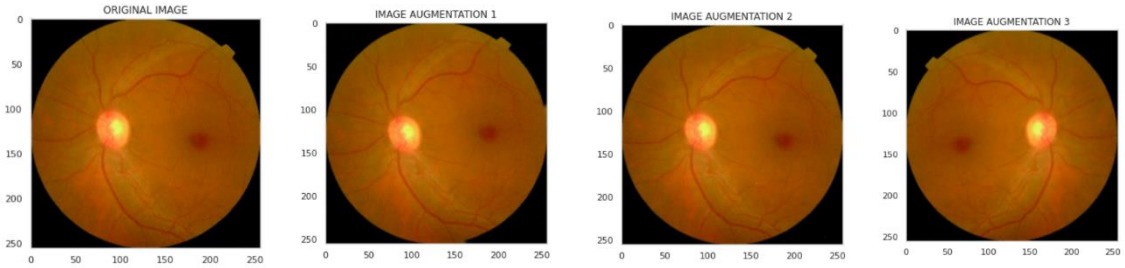

Figure 6. Data Augmentation

4.3 Convolutional Neural Networks

CNN is typically composed of three types of layers namely, convolution, pooling, and fully connected layers. The step where input data are transformed into output through these layers is called forward propagation. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others [5].

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of convolution operation and activation function. The outputs of a convolution are then passed through a nonlinear activation function. A pooling layer provides a typical down sampling operation which reduces the in-plane dimensionality of the feature maps to introduce a translation invariance to small shifts and distortions and decrease the number of subsequent learnable parameters. It is of note that there is no learnable parameter in any of the pooling layers, whereas filter size, stride, and padding are hyperparameters in pooling operations, like convolution operations.

The most popular form of pooling operation is max pooling, which extracts patches from the input feature maps, outputs the maximum value in each patch, and discards all the other values. The output feature maps of the final convolution or pooling layer is typically flattened, i.e., transformed into a one-dimensional array of numbers (or vector), and connected to one or more fully connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight. The crucial part of the deep convolutional neural network is handling the parameters generated from each stacked layer abundantly. It may cause over-fitting. For avoiding such scenarios, dropping out some neurons in the layer is beneficial [5].

For our problem, we have designed a basic CNN having three Convolution layers with ReLU activation function and Max Pooling operation with kernel size 2 and stride size 2. This is followed by four fully connected layers with dropout in between to avoid overfitting. The total number of parameters in the CNN is 769637 for image size 64*64 and 115,457,637 for image size 512*512. 2 different configurations of neural network are rendered as shown in Table 2.

Table 2. CNN Network Structure

Configuration	Image size	Epochs	Description	Statistics
1	64*64	10	Conv2D	3 layers
			Linear Layers	4 layers
			Output Layer	5 outputs
			Activation function	ReLU
			Max Pooling	2*2
			Dropout	3
			Optimizer	Stochastic Gradient Descent
2	512*512	10	Conv2D	3 layers
			Linear Layers	4 layers
			Output Layer	5 outputs
			Activation function	ReLU
			Max Pooling	2*2
			Dropout	3
			Optimizer	Stochastic Gradient Descent
			Augmentation	Horizontal Flip

4.4 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. It is an optimization that allows rapid progress or improved performance when modelling the second task. A pre-trained model has been previously trained on a dataset and contains the weights and biases that represent the features of whichever dataset it was trained on. Learned features are often transferable to different data.

For example, a model trained on a large dataset of bird images will contain learned features like edges or horizontal lines that you would be transferable your dataset. Pre-trained models are beneficial to us for many reasons. By using a pre-trained model, you are saving time. Someone else has already spent the time and compute resources to learn a lot of features and your model will benefit from it. For this dataset Resnet34 is used as the pre-trained model for transfer learning.

Resnet34 is a 34-layer convolutional neural network (Figure. 8) that can be utilized as a state-of-the-art image classification model. This is a model that has been pre-trained on the ImageNet dataset--a dataset that has 100,000+ images across 200 different classes. However, it is different from traditional neural networks in the sense that it takes residuals from each layer and uses them in the subsequent connected layers. Four different configurations of neural network are rendered as shown in Table 3.

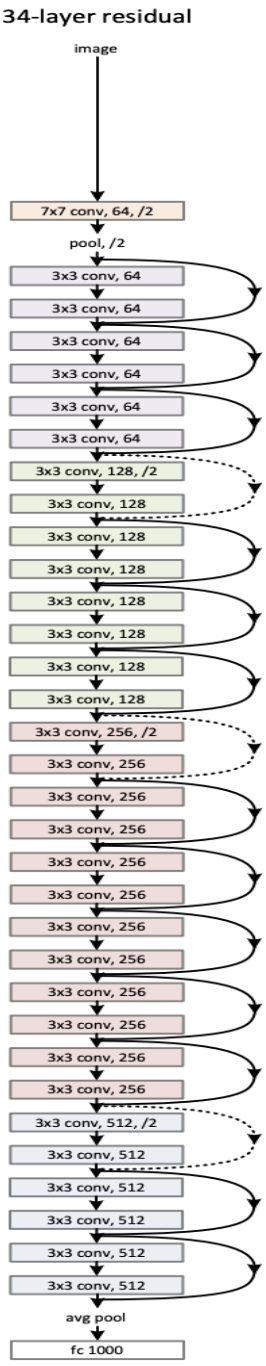

Figure 8. Example network architectures for ImageNet ResNet with 34 layers (3.6 billion FLOPs).

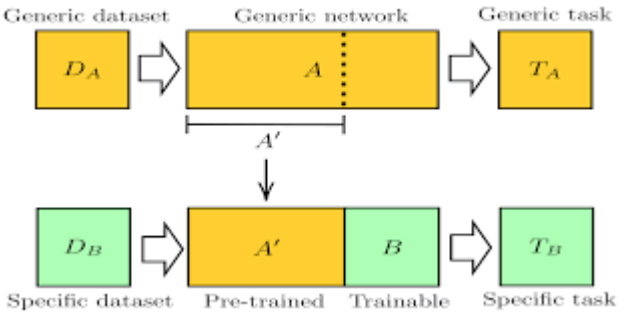

Figure 7: Transfer learning Architecture

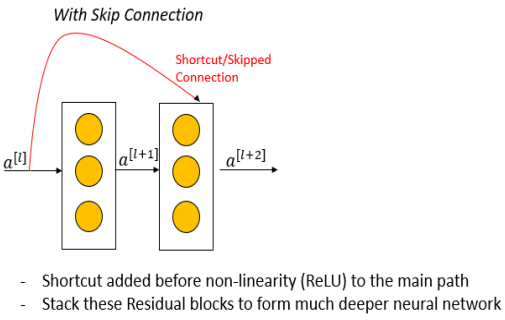

Figure 9. Residual Network Architecture

The resnet34 model is frozen and further linear and dropout layers are added to the model. Stochastic Gradient Descent (SGD) optimizer and cross entropy loss are used for training, the batch size of training data was set to 32 and a learning rate of 0.001 is used for all the 4 configurations.

Table 3. ResNet34 Network Structure

Configuration	Image size	Epoch	Description	Statistics
1	64*64	10	Resnet34	34 layers
			Linear Layers	3 layers
			Output Layer	5 outputs
2	64*64	10	Resnet34	34 layers
			Linear Layers	3 layers
			Drop out Layer	2 layers
			Output Layer	5 outputs
3	512*512	10	Resnet34	34 layers
			Linear Layers	3 layers
			Output Layer	5 outputs
4	512*512	30	Resnet34	34 layers
			Linear Layers	3 layers
			Drop out Layer	2 layers
			Output Layer	5 outputs

4.5 Multi-task Learning Technique

Multi-task learning is gaining popularity for image classification now-a-days because this type of classification involves high correlation between tasks. Multi-task learning is a machine learning technique in which we attempt to learn many tasks at the same time while minimizing numerous loss functions. We allow a single model to learn to accomplish all the tasks at once, rather than training separate models for each activity. The model learns generalized representations of the data that are useful in several situations by combining all the available data from the various tasks. Each task will have its own loss function, called L_i. We simply weight each loss function in our multi-task model and minimize the sum of these weighted losses [7].

$$\min_{\theta} \sum_{i=1}^{T} w_i \mathscr{L}_i(\theta, \mathscr{D}_i)$$

For this model, we perform data pre-processing and augmentation and save the processed images in a separate folder and use that for detection. The images are reduced to 512*512. For better classification, we convert multi-class labels to a binary matrix and perform cosine learning rate decay as the first few epochs will have a larger error rate and the training done after those will only increase the number of epochs which in increases processing time. To prevent this, we use a warm-up factor which increases the learning rate from 0 to a certain value by a factor during a period.

The training will begin after the epoch has reached a certain value. To perform multiple tasks, we use ResNet50 as our base model and perform regression, classification, and ordinal regression by using linear and SoftMax activation functions. All these tasks are concatenated, and the final decision is applied after that. The multi-task learning architecture in Figure 10.

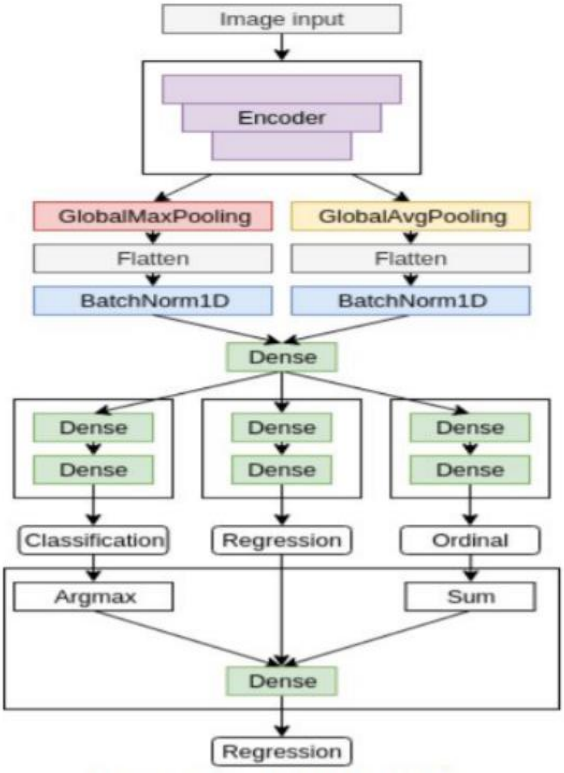

Figure 10. Multi-task learning architecture

We perform data pre-processing and data augmentation before we process our tasks on the data. We consider the image size to be 512*512 for pre-processing for this approach. We produce 3 images for a single image by data augmentation techniques and store these images. The batch size is 32. For better processing, we convert the multi-class labels to a binary matrix. We take the base class to be ResNet50 and apply regression, classification and ordinal regression as given in Figure 7. The step size of the training data is 91 and that of the validation data is 22. The step-by-step fit involves reducing loss, imputing warm-up epochs and so on.

Table 4. Multi-task learning Network Structure

Configuration	Image size	Epoch	Description	Activation Layer
1	512*512	25	Regression	Linear
			Classification	Softmax
			Ordinal Regression	Softmax
2	512*512	5	Regression	Linear
			Classification	Softmax
			Ordinal Regression	Softmax
3	512*512	25	Regression	Linear
			Classification	Softmax
			Ordinal Regression	Softmax

We also apply binary and categorical loss and use cross-entropy while applying categorical loss. At each and every fitting step. We write back the h5 file with new inferences and information about the predictions. Post-training, we use the Sequential model with a dense layer with linear activation. The learning rate for the entire modelling is assumed to be 1e-4 * 4 and the number of warmup epochs is taken as 5. All three layers have an output space of 2048 with batch size as 32, height as 320, width as 320 and canal as 3. We apply categorical focal loss 'SoftMax' to see the probability of the results to reduce class imbalance. The method is used as it generalizes well on the unseen data and prevents overfitting [8].

4.6. Computing environment

We ran all our models using GPU in Python 3 Google compute engine backend - Google Colab - RAM 12.69 GB and Kaggle – RAM 13 GB, GPU Memory – max 15.9 GB for all the above-mentioned models.

5. Results

This project proposed a DR classification system using multiple deep learning techniques including basic CNN, Transfer Learning using ResNet34 and Multi-task transfer learning. The proposed classification system provides classification for DR images to help ophthalmologists diagnose the patients' DR stage. The experimental results demonstrated that our custom ResNet34 model achieved best classification results of the various methods employed.

5.1 Main results

5.1.1 Convolution Neural Networks Results

We trained a simple CNN model using various image, batch, and epoch sizes on Kaggle and Google Colab with and without applying pre-processing and image augmentation techniques on 3662 images. The model was trained on Google Colab GPU, for 10 epochs with images of size 64*64 and with batch size 64. The best test accuracy came out to be about 52.2%. The model trained on GPU on Kaggle, for 10 epochs with images of size 128*128 and with batch size 64. The best test accuracy came out to be about 59.7%. The model trained on Kaggle using GPU, for 10 epochs with 3662 images of size 512*512 and with batch size 32 gave test accuracy of 68.3% with no pre-processing of image, but with horizontal flip and dropout.

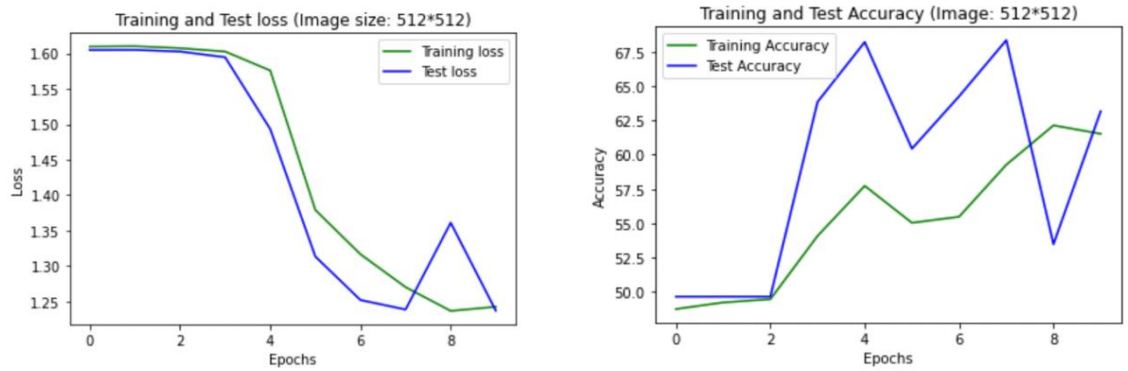

Figure 11. Loss curve and Accuracy curve for Configuration 2 [Image size: 512*512]

	severity	True Positive	False Negative	False Positive	True Negative	Precision	Recall	F1-score
0	0	16	4	2	84	0.888889	0.8	0.842105
1	1	12	8	16	88	0.428571	0.6	0.500000
2	2	10	10	21	90	0.322581	0.5	0.392157
3	3	4	16	19	96	0.173913	0.2	0.186047
4	4	0	20	0	100	NaN	0.0	NaN

Figure 12. Metrics of configuration 2 [Image size: 512*512]

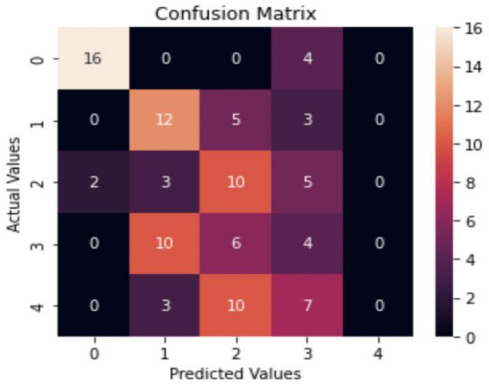

Figure 13. Confusion Matrix of 2 [Image size: 512*512]

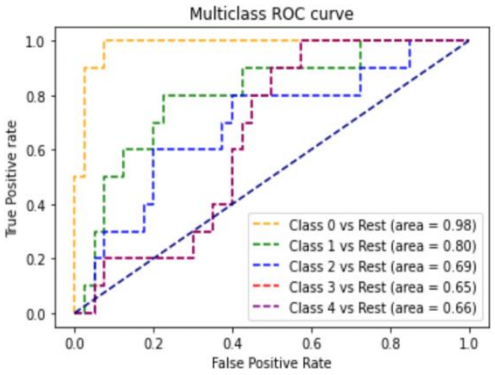

Figure 14. AUC-ROC Curves for 2 [Image size: 512*512]

After this, we trained models with image pre-processing including resizing, cropping, applying gaussian blur and circle crop and augmentation techniques like rotation, horizontal flip, and vertical flip. The model was trained on Kaggle GPU, for 17 epochs with 3662 images of size 512*512 and with batch size 32 gave best test accuracy came of 59.2%. We removed the circle crop and tried again, and the best test accuracy came out to be 59.6% (for 16 epochs) and 56.2% (for 30 epochs) with and without dropout respectively. So, dropout seemed useful.

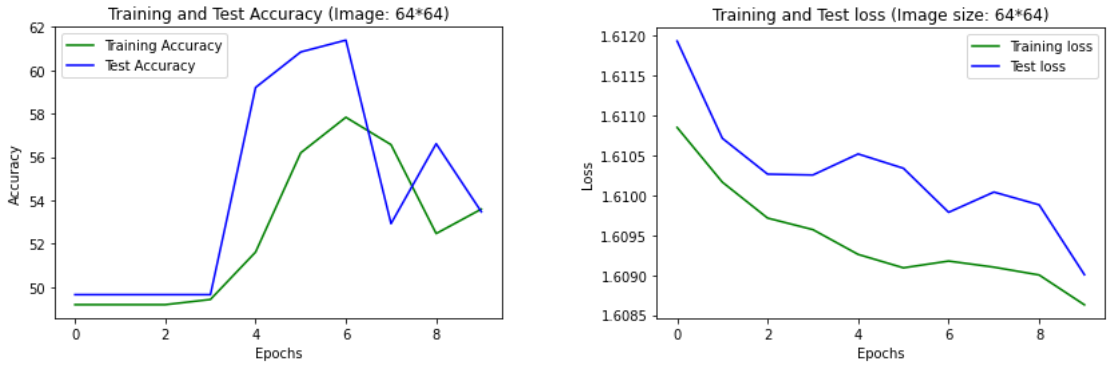

Figure 15. Loss curve and Accuracy curve for Configuration 1 [Image size: 64*64]

Another model was trained on Google Colab, for 10 epochs with 3662 images of size 64*64 with batch size 32 and this gave best test accuracy of 49.6%, with horizontal flip and without dropout and achieved 61.3% with dropout and no augmentation applied.

	severity	True Positive	False Negative	False Positive	True Negative	Precision	Recall	F1-score
0	0	18	2	24	82	0.428571	0.9	0.580645
1	1	12	8	34	88	0.260870	0.6	0.363636
2	2	0	20	8	100	0.000000	0.0	NaN
3	3	0	20	4	100	0.000000	0.0	NaN
4	4	0	20	0	100	NaN	0.0	NaN

Figure 16. Metrics of configuration 1 [Image size: 64*64]

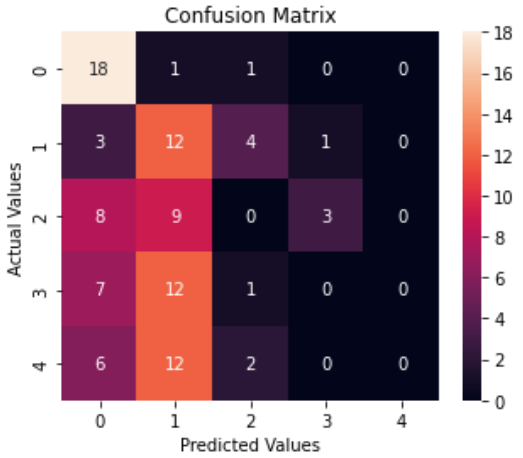

Figure 17. Confusion Matrix of 1 [Image size: 512*512]

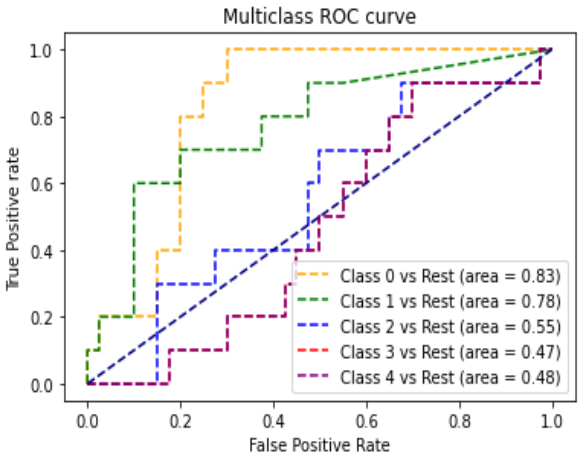

Figure 18. AUC-ROC Curves for 1 [Image size: 512*512]

The overall results of all configurations tried show that due to imbalance in data and more data in class 0, the models seem to perfectly predict images with severity 0. Severity class 4 has not been predicted well in both cases portrayed. Also, it can be seen that the model with dropout improves the accuracy. We can see that class 0 has the highest AUC and the least being class 4 in both the configurations.

5.1.2 Transfer Learning Results

Four different configurations of transfer learning model are trained, and the following metrics were recorded

- Train loss and Test loss
- Train accuracy and Test accuracy
- Confusion Matrix - 20 images per class
- Precision, Recall, F1Score – extended for Multiclass
- AUC (Area Under the Curve) - extended for Multiclass
- ROC (Receiver Characteristic Operator) Curve - extended for Multiclass

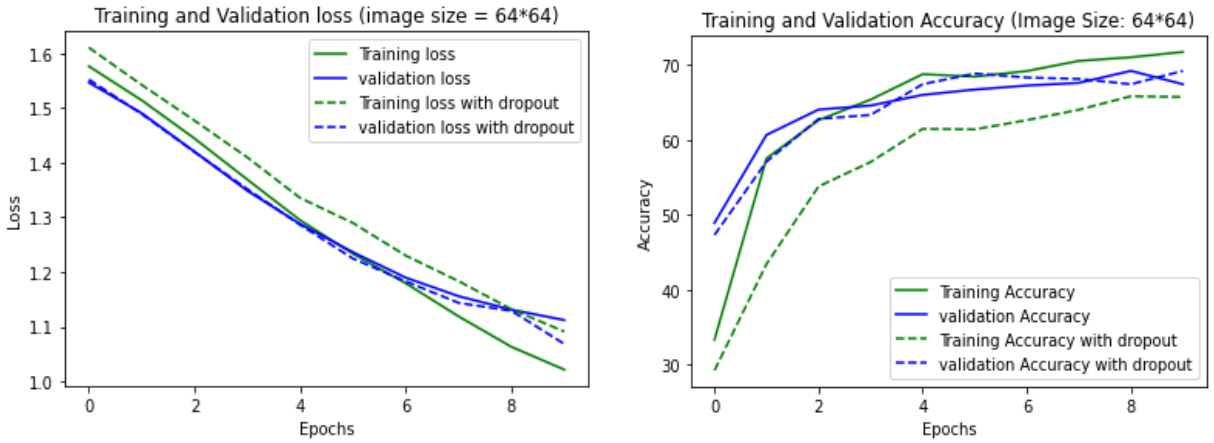

Figure 19. Loss curve and Accuracy curve for configurations 1 and 2 [Image size: 64*64]

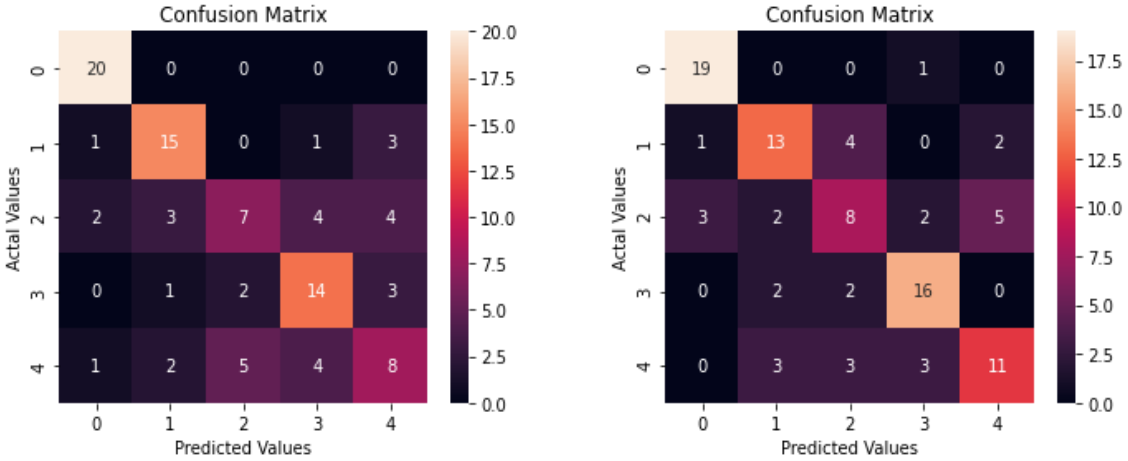

Figure 20. Confusion Matrix of configuration 1(left) and 2(right) [Image size: 64*64]

	severity	True Positive	False Negative	False Positive	True Negative	Precision	Recall	F1-score
0	0	19	2	1	81	0.950000	0.904762	0.926829
1	1	9	12	3	91	0.750000	0.428571	0.545455
2	2	17	4	24	83	0.414634	0.809524	0.548387
3	3	4	12	1	96	0.800000	0.250000	0.380952
4	4	9	12	13	91	0.409091	0.428571	0.418605

	severity	True Positive	False Negative	False Positive	True Negative	Precision	Recall	F1-score
0	0	19	1	4	81	0.826087	0.95	0.883721
1	1	13	7	7	87	0.650000	0.65	0.650000
2	2	8	12	9	92	0.470588	0.40	0.432432
3	3	16	4	6	84	0.727273	0.80	0.761905
4	4	11	9	7	89	0.611111	0.55	0.578947

Figure 21. Metrics of configuration 1 (top) and 2 (bottom) [Image size: 64*64]

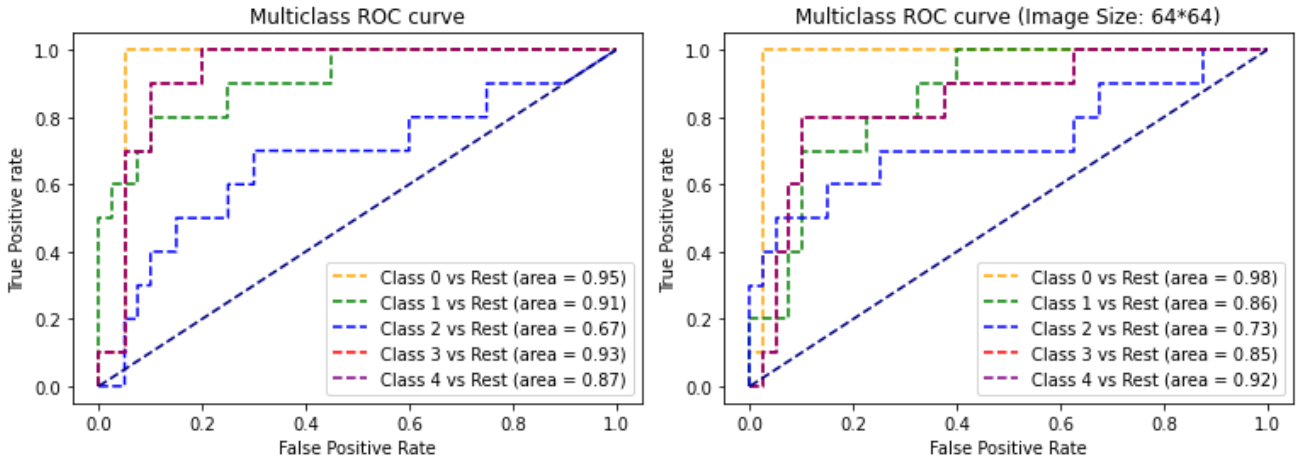

Figure 22. AUC-ROC Curves for configurations 1(left) and 2(right) [Image size: 64*64]

The overall results of configurations 1 and 2 show that due to imbalanced data and huge number of data being available for severity 0 the models seem to perfectly predict images with severity 0. Also, it can be seen that the model with dropout (configuration 2) performs marginally better than the other model (configuration 1).

For image with dimension 64*64 the configuration 2 with dropout produces best result with high accuracy and least loss even though the train loss for configuration 1 is less the test loss for configuration 2 is better. One interesting finding is that network without dropout predicts severity 2 better than with dropout.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. We can see that class 0 has the highest AUC and the least being class 2 in both the configurations.

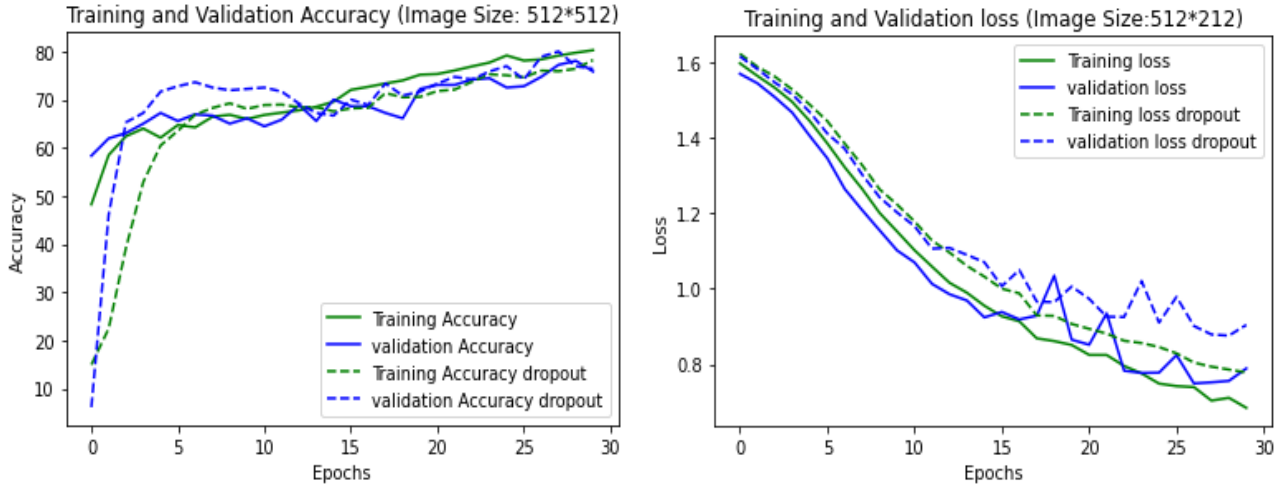

Figure 23. Loss curve and Accuracy curve for configurations 3 and 4 (Image size: 512*512)

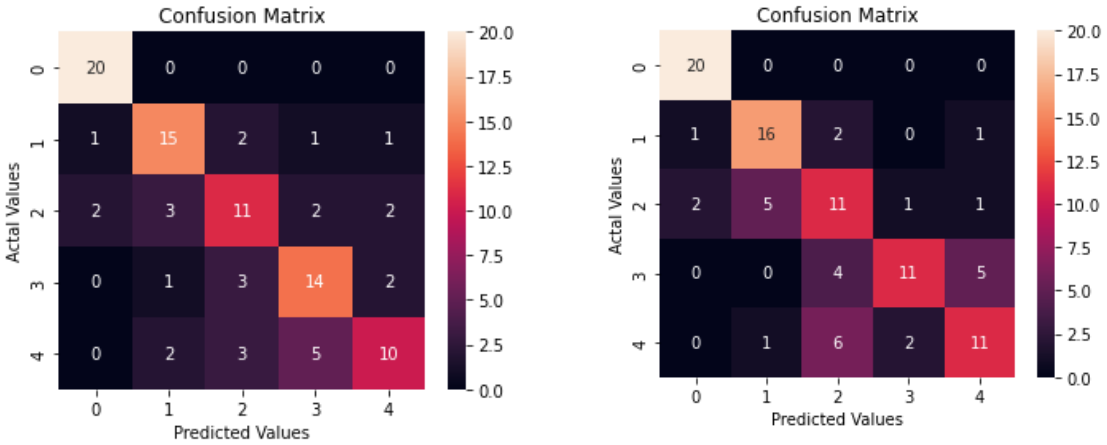

Figure 24. Confusion Matrix of configuration 3 and 4 (Image size: 512*512)

	severity	True Positive	False Negative	False Positive	True Negative	Precision	Recall	F1-score
0	0	20	0	3	80	0.869565	1.00	0.930233
1	1	16	4	6	84	0.727273	0.80	0.761905
2	2	11	9	12	89	0.478261	0.55	0.511628
3	3	11	9	3	89	0.785714	0.55	0.647059
4	4	11	9	7	89	0.611111	0.55	0.578947

	severity	True Positive	False Negative	False Positive	True Negative	Precision	Recall	F1-score
0	0	20	0	3	80	0.869565	1.00	0.930233
1	1	15	5	6	85	0.714286	0.75	0.731707
2	2	11	9	8	89	0.578947	0.55	0.564103
3	3	14	6	8	86	0.636364	0.70	0.666667
4	4	10	10	5	90	0.666667	0.50	0.571429

Figure 25. Metrics of configuration 3(top) and 4(bottom) [Image size: 512*512]

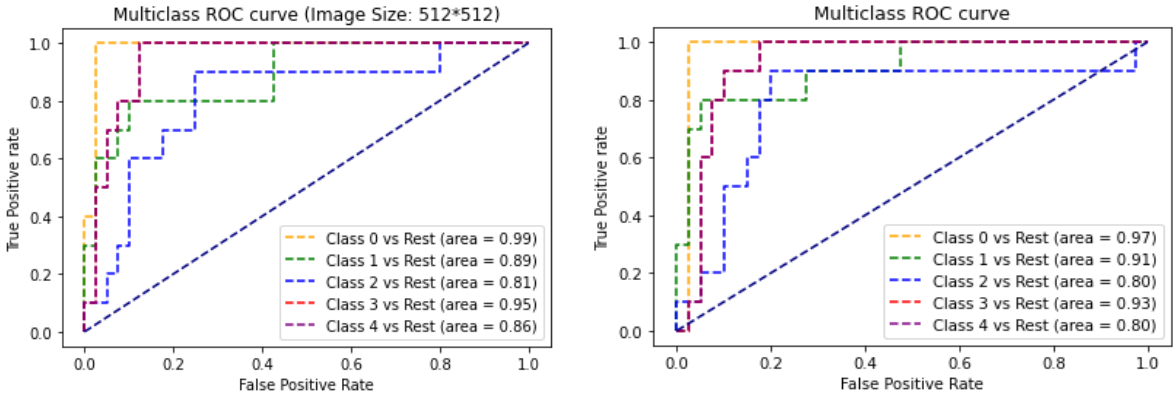

Figure 26. AUC-ROC Curves for configurations 3 and 4 (Image size: 512*512)

The overall results of configurations 3 and 4 show that both models perform better than configurations 1 and 2. This could be attributed to the increase in the dimension of the image from 64*64 to 512*512 pixels thereby making the model more robust. Also, it can be seen that the model without dropout (configuration 3) performs marginally better than the other model (configuration 4) both on training and test data.

An Interesting finding is that the configuration 4 performs better in the predicting class 3 and nearly equals the prediction accuracy in terms of predicting class 2 and class 1. The AUC - ROC curve also looks similar. Hence, we can conclude that dropout layers have very little impact on the model as a whole.

5.1.3 Multi-task learning Results

We use the following metrics for evaluation:

1. Regression task is evaluated using mean absolute error
2. Categorical loss is used for classification
3. Binary cross entropy loss is used for ordinal regression

The accuracy obtained with the methods before freezing the base layer (epochs = 25) is:

1. 37% for regression
2. 65% for classification
3. 77% for ordinal regression

The loss values we obtain for the above constraints above are:

1. 1.152 for regression
2. 1.0347 for classification
3. 0.4274 for ordinal regression

```
==========================================================================================
==============
Total params: 48,798,603
Trainable params: 48,737,291
Non-trainable params: 61,312
__________________________________________________________________________________________
______________
```

Figure 27. Parameter information for the first process

The accuracy of the models obtained by freezing the base layer and applying binary and categorical focal losses (epochs = 5) is:

1. 37% for regression
1. 66% for classification
1. 79% for ordinal regression

The loss values we obtain for the above constraints above are:

1. 2.2714 for regression
2. 0.0410 for classification
3. 0.1624 for ordinal regression

```
==========================================================================================
==============
Total params: 48,798,603
Trainable params: 25,202,699
Non-trainable params: 23,595,904
__________________________________________________________________________________________
______________
```

Figure 28. Parameter information for the second process

After unfreezing of the base layer and applying binary and categorical focal losses (epoch = 25), the accuracy of the models is:

1. 37% for regression
2. 66% for classification
3. 79% for ordinal regression

The loss values we obtain for the above constraints above are:

1. 2.1837 for regression
2. 0.0419 for classification
3. 0.1624 for ordinal regression

```
==========================================================================================
==============
Total params: 48,798,603
Trainable params: 48,737,291
Non-trainable params: 61,312
__________________________________________________________________________________________
______________
```

Figure 29. Parameter information for the second process

From this, we understand that freezing of layers does not affect the model's performance.

5.1.4 Comparison

Table 5. Comparison of various models

Configuration	Image size	Accuracy %
ResNet34	512*512	80.00
Multi-task learning	512*512	79.63
CNN	512*512	68.35

5.2 Supplementary Results

We have chosen ReLU activation function [15] and Max Pooling operation with kernel size 2 and stride size 2 since the size of the pooling operation or filter is smaller than the size of the feature map and is applied on each layer and we wanted to capture the minute details in the retina images. ReLU is chosen because it increases the non-linearity in our images, converges faster, is cheap to compute and is sparsely activated i.e., does not activate all neurons at the same time [17]. We used Stochastic Gradient Descent (SGD) optimizer and cross entropy loss for the CNN model with an 80-20 train and test data split. SGD optimizer can generalize well and thus improves final performance [12]. Cross entropy loss is typically used for multi class classification. It calculates a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem [14]. We have chosen cross entropy loss because it minimizes the distance between actual and predicted.

The hyperparameters chosen for the multi-task learning are 512*512 for image size, a batch size of 32 and warmup epochs for cosine decay of 5. We used linear activation for regression and SoftMax activation functions for classification and ordinal regression. SGD optimizer is also utilized here. We used mean squared error as the regression metric because it determines how close estimations or projections are to actual values [18]. The lower the MSE, the more accurate the prediction. We use focal loss for both classification and ordinal regression as we have a highly imbalanced dataset.

6. Discussion

We started off with a basic CNN model having dropout using image size of 64*64 without any pre-processing and augmentation. The best test accuracy came around 59 percent. We used batch sizes of 64 initially and then changed to 32. And we tried without dropout as well all for 10 epochs and it didn't go beyond 50 percent. We tried pre-processing and data augmentation for the image size 512*512 both with and without dropout and the best test accuracy obtained was around 59 percent for 30 epochs. The best result was obtained when we tried for 10 epochs without pre-processing but with horizontal flip and dropout and the accuracy obtained was 68.3 percent.

But as expected the transfer learning approach using Resnet34 was able to outperform CNN by a large margin with an accuracy rate of 80%. Even with less image dimension of 64*64 pixel the model was able to capture more features that traditional CNN. The cross-entropy loss was also least for this model, and it was able to capture majority of the severity class information with high precision.

Multi-task learning gets a slightly better accuracy when compared to other approaches which use several tasks. This might be because our approach involves data pre-processing steps followed by data augmentation. As the number of training samples increases, the model works better. By that notion, our model also performs better in that circumstance. By using warmup epochs, we have also resolved the issue of having lower accuracy in the first few steps. Our approach also uses a denser image size (256) as compared to other models. The highest accuracy we get is approximately 80%. This approach can be made much better by applying a greater number of heavy tasks on top of each other like stacking and use highly computational transfer learning architectures such as the EfficientNet series as the base model.

The top solution available solution on the Kaggle competition [19] gives an accuracy of 84% on private dataset. The approach used was ensemble of eight models - 2 x inception_resnet_v2, 2 x inception_v4, 2 x seresnext50, 2 x seresnext101 and taking the average. This work shows that the existing transfer learning model can be further tuned using other pre trained deep learning networks to improve the accuracy.

7. Conclusion & Future Work

The prevalence of diabetes is increasing worldwide, and the complication of DR is also on the rise. This disorder is threatening to diabetes patients if vision if DR is detected in the later stages. Therefore, the detection and treatment of DR in its initial stages is essential to decrease the risk of blindness. The manual diagnosis process of DR with the increasing suffering from DR is not much effective. Recently, the deep learning (DL) method has achieved superior performance in image classification. The current work provides an effective system to help in

DR diagnosis. In the future, we aim to combine multiple datasets to achieve a proper balance of the dataset. Since there was a limit on computation power, we were not able to run using large datasets. We would also like to improve on the current models implemented with better hyperparameter tuning, image pre-processing and data augmentation techniques.

8. Member Contributions

Joann conducted experiments using CNN, Manasa worked on multi-task learning techniques using ResNet50 as the base model and Sree Krishna worked on Transfer Learning using ResNet34. All worked on image pre-processing and data augmentation techniques required for the models and implemented the various metrics for evaluation of the models as well.

GitHub link: https://github.com/Sree-Krishna/APTOS-2019-Blindness-Detection

References

[1] Silberman, Nathan & Ahrlich, Kristy & Fergus, Rob & Subramanian, Lakshminarayanan. (2010). Case for Automated Detection of Diabetic Retinopathy.
[2] Pratt, Harry & Coenen, Frans & Broadbent, Deborah & Harding, Simon & Zheng, Yalin. (2016). Convolutional Neural Networks for Diabetic Retinopathy. Procedia Computer Science. 90. 200-205. 10.1016/j.procs.2016.07.014.
[3] Tymchenko, Borys & Marchenko, Philip & Spodarets, Dmitry. (2020). Deep Learning Approach to Diabetic Retinopathy Detection.
[4] Wejdan L. Alyoubi, Maysoon F. Abulkhair& Wafaa M. Shalash (2021). Diabetic Retinopathy Fundus Image Classification and Lesions Localization System Using Deep Learning.
[5] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do & Kaori Togashi (2018). Convolutional Neural networks: an overview and application in radiology.
[6] Mohammad T AL-Antary & Yasmine Arafa (2021). Diabetic Retinopathy Classification using Deep Learning framework.
[7] https://towardsdatascience.com/multi-task-learning-in-machine-learning-20a37c796c9c
[8] https://arxiv.org/pdf/2003.02261.pdf
[9] https://en.wikipedia.org/wiki/Gaussian_blur
[10] https://www.kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy
[11] Kaggle dataset: https://www.kaggle.com/c/aptos2019-blindness-detection/
[12] https://mlfromscratch.com/optimizers-explained/#/
[13] https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008
[14] https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/
[15] https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
[16] https://care.diabetesjournals.org/content/27/suppl_1/s84
[17] https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7
[18] https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/
[19] https://www.kaggle.com/c/aptos2019-blindness-detection/discussion/108065#latest-624210