Transfer Learning based Classification of Diabetic Retinopathy Stages

Adarsh Pradhan[1], Bhaskarjyoti Sarma[2], Rahul Kumar Nath[3] and Bhiman Kr Dey[4]

[1]*Assistant Professor*
Department of Computer Science and Engineering
Girijananda Chowdhury Institute of Management and Technology, Azara, Guwahati - 781017, Assam, India
adarsh@gimt-guwahati.ac.in

[2,3,4]*Department of Computer Ssience and Engineering*
Girijananda Chowdhury Institute of Management and Technology, Azara, Guwahati - 781017, Assam, India
{bhaskarjyotisarma85, rahulnath847 & bhimandey}@gmail.com

***Abstract* - Diabetic Retinopathy is caused due to damaging the blood vessels located at the back of the eye (retina). Symptoms such as blurriness, dark spots, blindness, etc. can be seen due to diabetic retinopathy. In this paper, we perform the analysis of diabetic retinopathy detection using three pre-trained networks – VGG16, InceptionV3, and ResNet50 on Kaggle Diabetic Retinopathy dataset. The dataset consists of five classes of images and is disproportionately balanced. So, we augmented new images, perform preprocessing, use different evaluation metrics and finally compared the results of these three pre-trained networks. Out of these three models, VGG16 achieves the highest accuracy of 78% and Kappa score of 0.721. We also observe that out of the five classes of images present in the dataset, class 2 has the lowest precision, recall as well as F1-score values.**

Index Terms - diabetic retinopathy, VGG16, InceptionV3, ResNet50.

I. Introduction

There are about 425 million people with diabetes in the world; about 82 million of the people are from South East Asia region, and by 2045 this will increase to 151 million [1][2][3][4]. While in India, there were over 72,946,400 cases of diabetes in 2017. Diabetic retinopathy (DR) damages the blood vessels within the retinal tissue which if left untreated, can lead to vision disability and blindness in older people [1]. The intricate case of DR is that in premature-stage there are no signs of DR, which makes it even more difficult to be detected in the its early stage. If a person has high levels of glucose then this could damage the blood vessels and limit the stream of blood flow to the retina. This vessel might scarcely leak blood without any distortion of vision; however, these vessels could be protected in its advanced stage. The eye will produce weak blood vessels that can easily break and leak into the vitreous gel of the eye. This bleeding blocks the retina and causes blurred vision, and could further lead to a detached retina and eventually a complete loss of vision. The risk of DR can be decreased if it is detected within time and with regular treatment. But many patients miss their best chance for treatment as the early stage of DR is symptomless [3]. Unfortunately the current practices used by experienced ophthalmologists for detection of DR are efficient but consumes time and fully depends on the practitioner [1]. Therefore, automated diagnoses within a short period of time and with more precision could be very beneficial for diabetic retinopathy detection.

In the past few years researchers have been giving much attention to automated detection of DR and several models were developed to increase the workability of DR screening which recognizes lesions like microaneurysms, haemorrhage, and exudates [4]. All the previous methods focus on two things; one is feature extraction and the other is prediction of DR [1]. The methods which use the above approaches (models like Decision Tree, SVM, Naive Bayes, kNN, K-Means, Random Forest) are efficient but also have various deficiencies. In recent times, methods like convolutional neural network (CNN) or deep convolutional neural network have been widely used which can automatically extract vital characteristics from the fundus images and also classify them [3][1].

In this work, our research has mainly focuses on the analyse of three pre-trained models, which are ResNet50, InceptionV3 and VGG16 on Diabetic Retinopathy Detection performed on Kaggle Diabetic Retinopathy dataset [8], which is a collection of five classes of Diabetic Retinopathy (DR) images, namely, class 0 or No DR, class 1 or Mild DR, class 2 or Moderate DR, class 3 or Severe DR, and class 4 or Proliferate DR. First we generated new images by using image augmentation for those classes which had fewer images, and then we apply image preprocessing techniques like cropping, resizing, Gaussian Blur, and max-min normalization. For cropping we have developed a method wherein we obtain four indexes at four positions of the image, and crop the image surrounding the four locations. To test the performance of our model we used Accuracy, Precision and Recall, F1-Score and Kappa Score. We see that out of these three models, VGG16 performs better than the rest of the two models.

II. Literature Review

Various researchers have already performed experiments on detecting diabetic retinopathy. Xiaoliang Wang et al [7] used Convolutional Neural Networks for classifying the Diabetic Retinopathy stage. 166 images were taken from the Kaggle dataset [8] to train their models. In their study, they have used transfer learning with three CNN, i.e., AlexNet, VGG16 and InceptionNetV3. After training the three models, the accuracy was found to be 37.43% for AlexNet, 50.03% for VGG16 and 62.23 for InceptionNet. Yashal Shakti Kanungo et al [9] used deep learning for detecting Diabetic Retinopathy. Preprocessing techniques such as normalization, mean subtraction and Principal Component Analysis (PCA) was performed on the dataset. The model used was based on InceptionV3 architecture. According to [9], even after training the model with a limited number of datasets, the model achieved a respectable score. Rahul Ghosh et al [6] used CNN for detecting the Diabetic Retinopathy stages. Images were used from the Kaggle dataset which are cropped and resized to 512 X 512 pixels, normalized and denoised. The accuracy achieved after training the model was found to be 85% and 95% for five class and two class classification respectively. Darshit Doshi et al [10] used deep CNN for detecting Diabetic Retinopathy. The images from the dataset were rescaled down to 512 X 512 pixels and the green channel is extracted. To make the CNN more robust and to prevent the CNN from reading the noise, the contrast of the images were enhanced and normalized. The best quadratic kappa score was found to be 0.3996. Sairaj Burewar et al [2] worked on detecting Diabetic Retinopathy by retinal segmentation with region merging using CNN.

In [11], they create a five-layered CNN model where the model is trained by using the ImageNet dataset, giving the accuracy of 98.15%. The same dataset is used for a comparative study among three pre-trained architecture - AlexNet, VGG-16 and SqueezeNet, and achieve an accuracy of 93.46, 91.82 and 94.49, respectively.

In another paper [12], where they did a comparative study between different architectures using Messidor datasets and by replacing the ANN classifier with SVM, they find out that the Resnet model gives the highest accuracy of 95.83% on Base-12 Messidor dataset containing 77 images. Inception-V3 and VGG-19 models provide the highest accuracy of 95.24% on the Base-13 Messidor dataset containing 70 images.

In another proposed method [13], where they have used two different pre-trained models, i.e., Inception-V3 (accuracy-87.12%) and Xception (accuracy-74.49%), and also have used various parameters like activation function (ReLu and ELU), optimizer (like Adam and SGD) against the model for comparative study.

III. Methods

A. Convolutional Neural Networks

Convolutional Neural Networks, also known as ConvNet and CNN, are commonly used in models that are trained for visual recognition systems as it can learn different patterns of the input using different filters. The model extracts more high-level features of the input as it goes deeper into the CNN layers.

CNN has many layers. The convolution layer uses different filters to extract the features from the input. Then the output from the convolution layer goes through the ReLU (Rectified Linear Unit) activation function which changes the pixel values to zero for the pixel values lesser than a certain threshold value and linear for values higher than that. After the activation function, it goes through the max-pool layer which extracts the maximum pixel value from the patch. Now the values are flattened and fed into a fully connected layer. Finally, it goes through another activation function, Softmax, which gives the probability distribution of the output classes.

B. Pre-Trained Models

1) VGG-16

In many research papers, VGG16, shown in Fig. 1, is used as a standard architecture for the classification of diabetic retinopathy detection [11][12]. In our proposed system, we are applying VGG16 on diabetic retinopathy datasets collected from the Kaggle website [8]. VGG neural network was designed and developed by Karen Simonyan and Andrew Zisserman in the year 2014 [14]. This architecture was submitted to the Large-Scale Visual Recognition Challenge 2014 (ILSVRC 2014) and achieved the accuracy of 92.7% with the error rate of 6.8% [12][14].

VGG16 architecture accepts a fixed input size of 224*224 RGB image, where it has a total of 138 million parameters [12][11][14]. The architecture comprises of 5 blocks of convolution layer followed by a max-pool layer after each block and at the end three fully connected layers with 4096, 4096, 1000 neurons respectively and one Softmax layer for classification. VGG16 architecture uses a very small kernel size i.e., 3*3, where after every convolutional layer, a non-linear operation is performed by a ReLU activation function. Every block contains at least two convolution layers and at most three convolution layers where the number of filters for convolution increases with the power of two starting from 64 to 512.

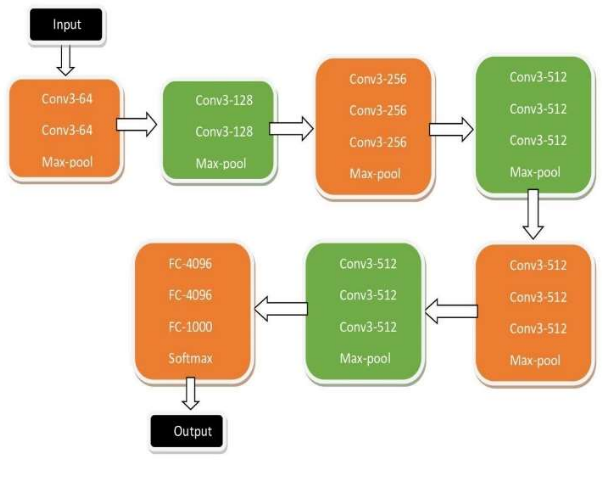

Fig 1: VGG16 Model

2) ResNet-50

The residual learning concept enables the neural network to go further into the deep layers in a neural network to extract more high-level features of images and used for classification with less error rate [15]. The ResNet neural network won the ILSVRC competition in 2015, where the ResNet model with 152 layers is trained using ImageNet datasets showing an error rate of 4.49% and 3.57% error rate on the ensemble model of the residual neural network [15]. The main idea behind the ResNet architecture is the skip connections or Identity connections, shown in Fig. 2, which helps the network to overcome the problem of vanishing gradients making the architecture to learn from more number of deeper layers [15]. In this study, we have used ResNet50 architecture, Fig. 3, with input size as 224×224×3 where channel width is 3. This architecture consists of 4 stages where at the initial steps it convolves the image with a 7×7 kernel, followed by the 3×3 max-pool layer. Every stage consists of several convolution blocks with identity connections where each block is having three convolutional layers with a kernel size of 3×3 and 1×1, and there is one residual block between every stage which also has its convolution layers. In the end, there is one average pooling layer followed by one fully connected layer having 1000 neurons to classify the diabetic retinopathy images.

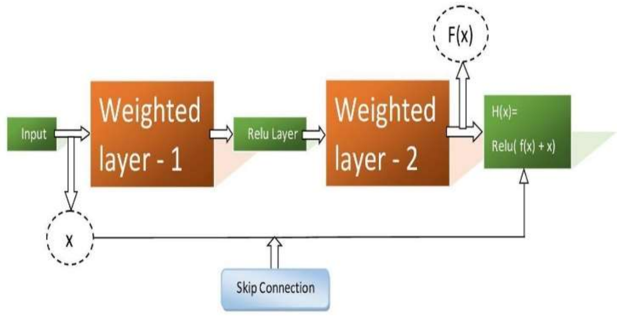

Fig 2: ResNet50 Model

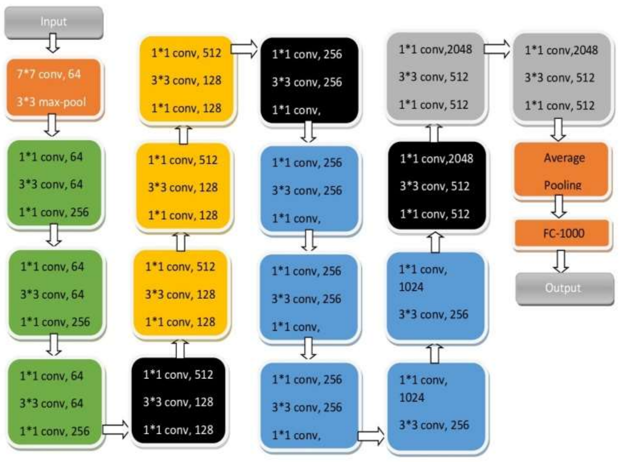

Fig 3: ResNet50 Model

3) *InceptionV3*

Inception-V3, Fig. 4, is a pre-trained 48 layers deep convolutional neural network which has been trained on over a million images by using the ImageNet database. The inception model was created as a part of LeNet project by Google. The Inception-V3 has the ability to classify upto 1000 categories of objects. The inception layer allows the internal layers to choose a relevant filter size to learn the necessary information. In the inception model, instead of having a single convolution, you have a composition of 1×1, 3×3 and 5×5 convolution and then you can simply concatenate the output of each of them. The output is then passed into the next layer [13]. An inception module looks like the one shown in the fig.4. Instead of having a single convolution, you have a composition of average pooling followed by a 1×1, then a 1×1 convolution, then a 1×1 followed by a 3×3, then a 1×1 followed by a 5×5. And at last, you simply concatenate the output of each of them.

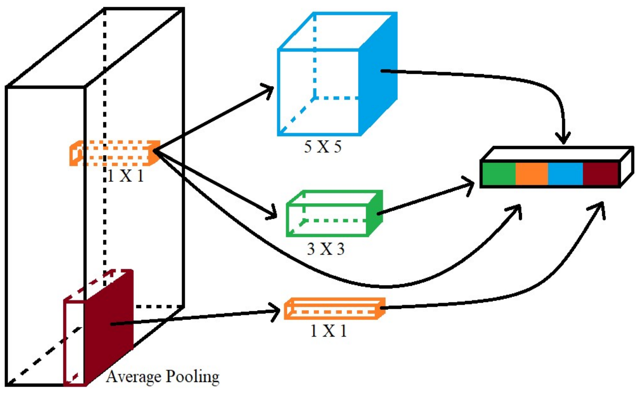

Fig 4: InceptionV3 Model

C. *Image Acquisition*

The dataset we have used here are fundus images from the Kaggle diabetic retinopathy dataset, which has 35,126 RGB retinal images. The images exhibit five stages of Diabetic Retinopathy (DR) as shown from Fig. 4 to Fig. 8. Each image has been clinician rated on a scale of 0 to 4, where, 0 represents No DR (Fig. 4), 1 represents Mild DR (Fig. 5), 2 means Moderate DR (Fig. 6), 3 means Severe DR (Fig. 7) and 4 represents Proliferative DR (Fig. 8).

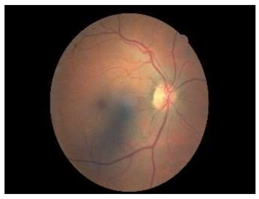

Fig. 4. No DR (Stage 1)

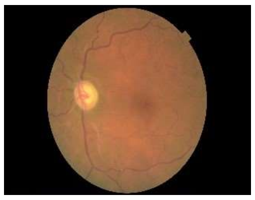

Fig. 5. Mild DR (Stage 2)

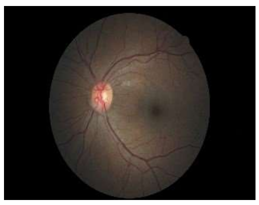

Fig. 6. Moderate DR (Stage 3)

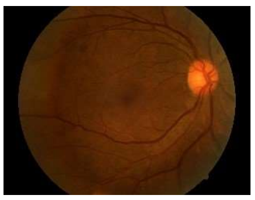

Fig. 7. Severe DR (Stage 4)

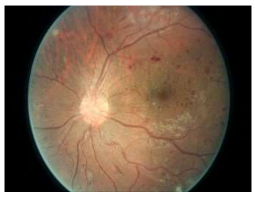

Fig. 8. Proliferative DR (Stage 5)

D. *Pre-processing*

The Retinal Fundus Images vary in various aspects such as image size, contrast, brightness, and many others. Also, the high-resolution images make it difficult in the learning process as they would require high computational power and training time.

Therefore, to reduce these complexities and to bring similarity in the image properties, we first need to prepare the dataset using some preprocessing steps. We have adopted cropping, resizing, Gaussian Blur, and max-min normalization illustrated below:

1) Cropping: The retinal images consist of black pixels padded around the retina. We have used a technique to remove these undesired pixels by calculating four indexes, which are finally used to crop the image. The procedure to obtain these four indexes is discussed below.

At first, we calculate and select the middle horizontal line of the image. We start by scanning every pixel one by one from the left end of the line, comparing the pixels RGB values with a particular offset (4 in our case). Whenever we find a pixel, having pixel value higher than the offset, we stop and record that pixel's index value as 'initial horizontal index'. Again, the same procedure is followed for 20 horizontal lines above and below the middle line. At the end, the smallest initial horizontal index is noted.

Again, we scan from the right end of the middle horizontal line and repeat the same process for 20 other lines above and below it. The highest index value from all these scans is noted as the 'final horizontal index'.

Similarly, 'initial vertical index' and 'final vertical index' are calculated using the middle vertical lines of the image by scanning from top and bottom respectively.

Once we obtain the four indexes at four positions of the image, we crop the image accordingly, discarding the unwanted black pixels.

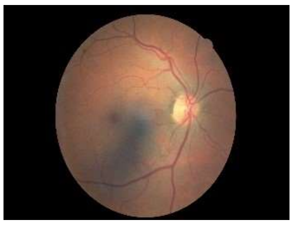
Fig. 9. Original Image

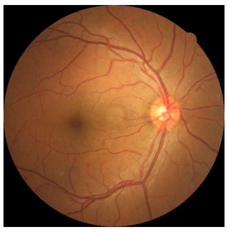
Fig. 10. Cropped Image

2) Resize: The dimensions of the images present in the dataset are not identical. Fitting the model with these dissimilar images is not possible. Therefore, to make the image dimension uniform, we resize them to a fixed size of 224 x 224 with the help of bilinear interpolation.

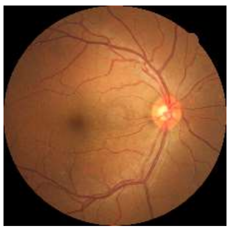
Fig. 11. Resized Image

3) Gaussian Blur: In order to enhance the image and to remove the noise we have used Gaussian blur. Gaussian blur can be applied to an image by convolving the image with a kernel that uses Gaussian function. The formula of a Gaussian function is given as (equation 1):

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

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.

We have used the Gaussian blur function from the OpenCV python library. We have set the kernel standard deviation along X-axis to 30 and Y-axis to zero. The Gaussian kernel size is computed from sigma values.

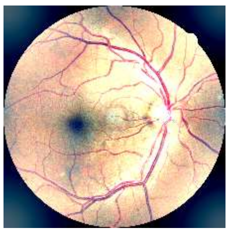
Fig. 12. Gaussian Blurred Image

4) Max-Min Normalization**:** Finally, before feeding the images to a model, we standardize the pixel values of the images to a scale of 0-1 using max-min normalization (equation 1).

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2}$$

x': normalized value, x : original value

E. *Training Algorithm:*

In this paper, we have employed Stochastic Gradient Descent with Nesterov Momentum as an optimization algorithm to train all three pre-trained models discussed above.

Stochastic Gradient Descent (SGD) [16] is a minor but most effective modification of the classical gradient descent (GD) algorithm. In SGD, at every epoch, the loss gradient is calculated from a random mini-batch or subset of the entire dataset, instead of calculating it using the entire dataset. Thus, speeding up the training time required per epoch at the cost of a noisier path to the minima. The introduction of momentum in SGD, which supports the persistent movement along the direction of improvement in the loss reduction, helps in the faster convergence towards the global minima by avoiding the local minima. In SGD with Nesterov Momentum, the gradient is computed at the position ($\theta_t + \mu v_{t-1}$) instead of the calculating at θ_t, therefore, regardless of the momentum term pointing in the wrong direction, the gradient always points in the right direction. The update rule is: (equation 3 and 4).

$$v_t = \mu v_{t-1} - \eta \nabla l(\theta_t + \mu v_{t-1}) \tag{3}$$

$$\theta_t = \theta_{t-1} + v_t \tag{4}$$

where,

't' → the number of iterations
'μ' → the momentum parameter
'∇' → the gradient
'l' → the loss function
'η' → the learning rate.

F. *Hyperparameters:*

Hyperparameters play the most significant role in machine learning. Even an insignificant change in the hyperparameters can bring a profound change to the whole learning process. Therefore, proper tuning of these hyperparameters is essential in bringing satisfactory results. We have adopted numerous experiments before finalizing the set of hyperparameters (Table 1) for every pre-trained model, especially the learning rate and the batch size. We have trained VGG16 and InceptionV3 for a total of 100 epochs with a batch size of 50. We set the learning rate for the initial 50 epochs to 0.001 and the remaining 50 epochs with 0.0001. We trained ResNet50 for 150 epochs with a learning rate of 0.0001 and a batch size of 100. We keep the learning rate decay and momentum co-efficient as 10^6 and 0.9, respectively, for all the three models. The update schedule for the learning rate is (equation 5):

$$L_t = L_{t-1} * \frac{1}{1 + (D * t)} \tag{5}$$

where,

L → Learning Rate
t → No. of batch iteration
D → Learning Rate Decay

G. *Transfer Learning:*

We have adopted transfer learning to train all the three pre-trained models by using the ImageNet weights. We have fine-tuned the models by removing the top layers and replaced them with some new layers, including a max pool layer followed by three consecutive dense layers with 512, 256, and 128 nodes, respectively. Finally, we have an output layer comprising of a dense layer of 5 nodes with Softmax activation function. Before starting the training process, we have set all the layers of the models to trainable.

TABLE 1 HYPERPARAMETERS VALUES

Hyperparameters	Values		
	VGG16	InceptionV3	ResNet50
Initial Learning Rate	0.001	0.001	0.0001
Learning Rate after 50 epochs	0.0001	0.0001	0.0001
Learning Rate Decay	10^{-6}	10^{-6}	10^{-6}
Momentum Co-efficient	0.9	0.9	0.9
Batch Size	50	50	100
Number of Epochs	100	100	150

IV. EXPERIMENT AND RESULT

We observed that in the Kaggle Diabetic Retinopathy dataset, that the class distribution of the training dataset is not uniform; there are 25810 images from class 0 (No DR), 2443 images from class 1 (Mild DR), 5292 images from class 2 (Moderate DR), 873 images from class 3 (Severe DR) and 708 images from class 4 (Proliferate DR). So we decided to perform image augmentation and generate 10,000 instances for every class except for class 0. For example there are 2443 images from class 1, so we generated 7557 new images for class 0, to make a total of 10,000 images. The same process is repeated for class 2, 3 and 4 as well, making a total of 10,000 images for each class. The augmented images are generated by using rotation, vertical flip and horizontal flip. We then randomly select 10,000 images from class 0 to make the final dataset for the experiment uniform with a total fundus image of 50,000. Next, we randomly split these 50,000 images into training, validation, and testing set comprising of 85%, 10%, and 5% from the total images. Using OpenCV library, an image processing open-source library, we have performed the necessary preprocessing steps mentioned above.

After finishing with the preprocessing, we have performed several experiments in order to obtain suitable results. Initially, the validation accuracy did not improve; instead, it shows fluctuation after every epoch. We then, kept on decreasing the learning rate until the problem started diminishing. Initially, we trained the models for 50 epochs and found that our models performed much better with the initial learning rate of 0.001, 0.001, 0.0001 for VGG16, InceptionV3, and ResNet50, respectively. The results further improved after another 50 epochs, when the learning rates of VGG16 and InceptionV3 were decreased to 0.0001 and 0.0001, respectively.

Finally, we have proceeded to carry out the training of all three pretrained models VGG16, InceptionV3, and ResNet50 with the respective ImageNet weights, using the training and validation set for 100,100 and 150 epochs.

Google Collaboratory, a free service provided by Google, where an individual can use the high-end Collab's hardware to run their Python code directly through their browser utilizing cloud computing. As we are working with three pre-trained models, training them one after another would take much time to complete our study. Thus, Google Collaboratory made it much easier and faster to train those pre-trained models separately and simultaneously with excellent computing power. We have availed the GPU runtime equipped with 8GDDR5 VRAM Tesla P4 having 2496 CUDA cores, 1 core 2 threads Intel(R) Xeon(R) CPU @ 2.00GHz, 13GB RAM and available disk space of 34GB for 12 hours to perform our research work. In this study, the Keras API, along with the TensorFlow library, provided us with all the necessary functions required to conduct our machine learning experiment comfortably.

Metrics Evaluation:

We have calculated the metrics such as Accuracy, Precision and Recall, F1-Score and Kappa Score by predicting on the test sets for all the three trained models.

Accuracy (equation 6) is one of the common evaluation metrics which gives us the idea about the proportion of true instances among the total number of instances examined. From our experiments, we see that the accuracy of ResNet50 and InceptionV3 are same, 73%, whereas the VGG16 gives us the highest accuracy value of 78% (Table 2, 3 and 4).

Precision and recall are other two evaluation metrics that are widely used. Precision, as shown in equation 7, gives us a clear picture about what percentage of predicted positives is actually true positive whereas as recall, given in equation 8, tells us about what fraction of actual positives is correctly classified. The computed precision and recall of our model are depicted in Table 2, 3 and 4. From the three experiments (Table 2, 3 and 4), we observe that for all the three models, both the precision and recall for class 4 (Proliferate DR) has highest value, whereas, the second highest precision value for all the three models is for class 3 (Severe DR) but the second highest recall value for ResNet50 is class 1 (Mild DR) and for InceptionV3 and VGG16 is class 0 (No DR). The lowest precision value for all the three models is class 2 (Moderate DR), whereas the lowest recall value for ResNet50 and InceptionV3 is class 2 (Moderate DR), and for VGG16 is class 3 (Severe DR). Upon further analysis, we see that the precision value for class 1, class 3 and class 4 is quite good, with class 3 and class 4 scoring above 0.90. Similarly, we monitor that the recall value for class 0, class 1 and class 4 is better with lowest value being 0.77. We find that class 2 has performed quite poorly in both the case of precision and recall.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{\mathrm{TP + FN}} \tag{8}$$

TABLE 2 EVALUATION RESULTS FOR RESNET50

ResNet 50	**Class 0**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
Precision	0.60	0.68	0.57	0.90	0.98
Recall	0.78	0.83	0.52	0.55	0.97
F1-Score	0.69	0.75	0.55	0.68	0.97
Testing Accuracy = 73% & Kappa Score = 0.663					

TABLE 3 EVALUATION RESULTS FOR INCEPTION V3

Inception V3	**Class 0**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
Precision	0.58	0.75	0.58	0.79	0.99
Recall	0.83	0.77	0.46	0.59	0.99
F1-Score	0.68	0.72	0.52	0.67	0.99
Testing Accuracy = 73% & Kappa Score = 0.660					

TABLE 4 EVALUATION RESULTS FOR VGG16

VGG16	**Class 0**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
Precision	0.68	0.80	0.59	0.91	0.99
Recall	0.89	0.81	0.64	0.57	0.98
F1-Score	0.77	0.80	0.62	0.71	0.99
Testing Accuracy = 78% & Kappa Score = 0.721					

Another evaluation metric that we have used is the F1-score (equation 9) which is the harmonic mean of Precision and Recall. For class 4 (Proliferate DR), we obtained an F1-score of 0.97, 0.99 and 0.99 for ResNet50, InceptionV3 and VGG16, respectively, whereas, class 1 (Mild DR) has the second highest and class 2 (Moderate DR) has the lowest F1-score for all the three models.

$$F1 - Score = 2 \times \frac{Precision \ \times Recall}{Precision + Recall} \tag{9}$$

With Cohen's Kappa score (equation 10), we compute the degree of agreement between two evaluators. Here the predicted values and the truth values are the two evaluators. With observed accuracy and predicted accuracy, we have computed the Kappa score of the three models and obtained a moderate level of agreement all three models. For ResNet50 we obtain a Kappa score of 0.663, for InceptionV3 we obtained 0.660 and for VGG16 we obtained the highest Kappa score value of 0.721. This shows that VGG16 has performed much better compared to ResNet50 and InceptionV3.

$$Kappa\ Score(K) = \frac{p_o - p_e}{1 - p_e} \qquad (10)$$

V. CONCLUSION

In this paper, we employed transfer learning for classification of various Diabetic Retinopathy Stages. We used ResNet50, InceptionV3 and VGG16 architecture as our pretrained models and used three fully connected layers at the end with 512, 216 and 128 nodes, respectively. Since the Kaggle Diabetic Retinopathy dataset is highly imbalanced, we tried to create a balance class distribution by generating new images by using rotation, vertical flip and horizontal flip augmentation techniques. Then we performed various preprocessing methods, like cropping, resizing, Gaussian Blur, and max-min normalization. Training, validation and testing were performed in Google Collaboratory, a free service provided by Google. After several experiments and observations, we found that the three models performed better when the learning rate is set to 0.0001. Finally, we assessed the three models with different metrics, namely, Accuracy, Precision and Recall, F1-Score and Kappa Score. After investigating the evaluation metrics for all the three models, we analyzed that the class 4 and class 3, which are the Proliferate DR and Severe DR images, has pretty high precision values. Class 1 (Mild DR) also has moderate precision values. Similarly, the recall value for class 0, class 1 and class 4 is also quite good. The F1-score for class 4 (Proliferate DR), for all the three models, is the highest, with class 1 (Mild DR) being the second highest. Class 2 which are Moderate DR images has the lowest precision, recall as well as F1-score values. The accuracy and Kappa score for all three models have been found to be of moderate value with VGG16 performing the best with an accuracy of 78% and Kappa score of 0.721. Thus we observe that the VGG16 has performed quite better compared to ResNet50 and InceptionV3 architectures.

ACKNOWLEDGMENT

This work is supported by Assam Science and Technical University, under TEQIP III program of Ministry of Human Resource Development, India, funded by World Bank.

REFERENCES

[1] Z. Wang and J. Yang, "Diabetic Retinopathy Detection via Deep Convolutional Networks for Discriminative Localization and Visual Explanation," Mar. 2017.

[2] S. Burewar, A. B. Gonde and S. K. Vipparthi, "Diabetic Retinopathy Detection by Retinal segmentation with Region merging using CNN," 2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS), Rupnagar, India, 2018, pp. 136-142.

[3] X. Zeng, H. Chen, Y. Luo and W. Ye, "Automated Detection of Diabetic Retinopathy using a Binocular Siamese-Like Convolutional Network," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5.

[4] S. Roychowdhury, D. D. Koozekanani and K. K. Parhi, "DREAM: Diabetic Retinopathy Analysis Using Machine Learning," in IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 5, pp. 1717-1728, Sept. 2014.

[5] P. Junjun, Y. Zhifan, S. Dong and Q. Hong, "Diabetic Retinopathy Detection Based on Deep Convolutional Neural Networks for Localization of Discriminative Regions," 2018 International Conference on Virtual Reality and Visualization (ICVRV), Qingdao, China, 2018, pp. 46-52.

[6] R. Ghosh, K. Ghosh and S. Maitra, "Automatic detection and classification of diabetic retinopathy stages using CNN," 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, 2017, pp. 550-554.

[7] X. Wang, Y. Lu, Y. Wang and W. Chen, "Diabetic Retinopathy Stage Classification Using Convolutional Neural Networks," 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, 2018, pp. 465-471.

[8] Diabetic retinopathy detection: identify signs of diabetic retinopathy in eye images, https://www.kaggle.com/c/diabetic-retinopathy-detection.

[9] Y. S. Kanungo, B. Srinivasan and S. Choudhary, "Detecting diabetic retinopathy using deep learning," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2017, pp. 801-804.

[10] D. Doshi, A. Shenoy, D. Sidhpura and P. Gharpure, "Diabetic retinopathy detection using deep convolutional neural networks," 2016 International Conference on Computing, Analytics and Security Trends (CAST), Pune, 2016, pp. 261-266.

[11] Mobeen-ur-Rehman, S. H. Khan, Z. Abbas and S. M. Danish Rizvi, "Classification of Diabetic Retinopathy Images Based on Customised CNN Architecture," 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 2019, pp. 244-248.

[12] D. U. N. Qomariah, H. Tjandrasa and C. Fatichah, "Classification of Diabetic Retinopathy and Normal Retinal Images using CNN and SVM," 2019 12th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 2019, pp. 152-157.

[13] S. Mohammadian, A. Karsaz and Y. M. Roshan, "Comparative Study of Fine-Tuning of Pre-Trained Convolutional Neural Networks for Diabetic Retinopathy Screening," 2017 24th National and 2nd International Iranian Conference on Biomedical Engineering (ICBME), Tehran, 2017, pp. 1-6.

[14] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv 1409.1556, Sep. 2014.

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778.

[16]Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning," In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13). JMLR.org, III–1139–III–1147, 2013.