Deep Learning Approach for Early Stage Detection of Diabetic Retinopathy

Sai Chethan Singu
Msc. Data Analytics
(School of Computing)
National College of Ireland
Dublin, Ireland
x18181937@student.ncirl.ie

***Abstract*— Diabetes is one of the most considerable chronic health issues in many countries around the world. It is very common for the diabetic people to get affected with disease called Diabetic Retinopathy because of high levels of blood sugar in retina for a long time. One of the big challenges with diabetic retinopathy disease is that it won't show early signs and symptoms. It is very important to avert the increase in severity as it will eventually damage the eye or can cause sight. Many research works have been done using machine learning to detect diabetic retinopathy so that further damage to the eye can be avoided. This study is to come up with the deep learning techniques to detect diabetic retinopathy in the early stages. The research work was carried out based on Convolutional Neural Network (CNN) using transfer learning and by performing data augmentation. The dataset for this study comprises of around 35k retina images of left and right eyes. The severity was categorized on a scale of zero from no diabetic retinopathy to four which is proliferative retinopathy.**

Keywords—Diabetes, Diabetic Retinopathy, Retina, Convolutional Neural Network, Transfer Learning, Data Augmentation.

I. INTRODUCTION

Diabetic retinopathy is the main disease in diabetic patients that can cause blindness. It is also proved to be one of the world's most common and serious eye disease that can cause blindness. The number of diabetic patients were increasing but there is unmet need for screening examinations. And the current process is complicated and requires a greater number of trained doctors to evaluate and review eye images leading to loss of subsequent meetings and chance of miscommunication. The other main problem is that lack of infrastructure to accommodate the large number of diabetic retinopathy (DR) patients. DR is a disease which has affect gradually and can led to severe damage which will affect for long period of time. DR has stages from mild to severe and if the DR is detected in early stages there are more chances of preventing from getting this worse.

These issues were addressed by using deep learning techniques. Disclosure of DR in its early stages is very crucial step in the process of diagnosis. Hence by using deep learning methodologies, automatic detection of stages in diabetic retinopathy can be done. It was categorized on a scale from 0 to 4 that is from no DR to proliferative DR. With the help of data augmentation, training dataset can be expanded artificially by flipping, cropping, zooming and also by changing brightness. Augmentation will allow the model to learn the features that are invariant to the image's location. This research focuses on transfer learning techniques to achieve the severity of DR and also comparison of different techniques implied in this process will be done.

II. RELATED WORK

Diabetic retinopathy is the most common disorder which can be found in diabetic patients. At the same time, it is difficult to examine the disease in terms of mass population. A research work proposed the automatic detection of the disease by using ensemble method along with cross validation. And achieved good accuracy. Two levels classification was done where first level eliminates noise and outliers and second level constitutes of clean data which was used for classification [1].

Identification of diabetic retinopathy is very important in initial stages of examination. There are different approaches made by different researchers for the early detection of disease. A study work was proposed by the author for predicting the severity of diabetic retinopathy. Adaboost algorithm has been used by the author so that bias can be reduced. Eight different image transformations were introduced for the augmentation of retina images which resulted in good performance of the prediction model [2].

Many research works have been done so far to predict the severity of diabetic retinopathy which can cause blindness in diabetic patients. One such work is done by using not only machine learning but also deep learning. Convolution neural network algorithm has been used for the classification of pre-trained model. High resolution images of retina are cropped into 128x128 pixels and performed data augmentation with different parameters before using Convnet [3].

As there will be no signs of symptoms, detecting diabetic retinopathy in its early stages is very important to protect the eyes from blindness. A research work has been done by using multi-channel CNN for the prediction of various stages in disease. The main purpose of using multi-channel GAN is to produce series of retina images reciprocal to scattered DR features and also minimize the dependency on labeled data. DR feature extractor was also used to reduce the noise and capture representative DR characteristics [4].

One of the reason main reason for the vision impairment in diabetic patients is due to damage of blood vessels and plays important role for the early detection. A research work proposes the detection of DR in the retina images with the help of multi layered perceptron. Nineteen features were extracted from the images to feed neural network for classification. Bayesian Regularization have been used for training the model where it achieved around 72% accuracy [5].

The main reason for the occurrence of DR is due to chronic diabetes. In order to address the problem of detecting DR, full CNN which is a deep learning model has been used. In this proposed neural network, it consists of 6 layers with ReLu as its activation function. It was proved that, the model trains faster than traditional neural networks. To overcome the outliers, weights were re-tuned in this model where it showed an accuracy of 91.66% which was proved to be high when compared to the other research works [6].

This research work was done to detect DR in its early stages so that vision loss can be prevented. Transfer learning was used to generalize the model along with CNN. Based fine-tuning, computational time was significantly reduced. Images were cropped to 224x224 for the purpose of training and testing and VGGNet architecture was used for the prediction of severity in DR [7].

DR is one of the commonly occurring disease in diabetic patients. In this paper [8], author proposes a model to predict severity of DR using multilayer perceptron with back propagation. Fuzzy C clustering otherwise known as k-means have been used for grouping data that belongs to different groups. 2 hidden layers have been used on a sample of 100 images. The model achieved an accuracy of 94.11% .

2% of the total population in the world is being affected by diabetic retinopathy at least in an average year. The early stages of this crisis are called Non-proliferative Diabetic retinopathy (NPDR). if this not taken care of properly this can lead to Proliferative Diabetic Retinopathy (PDR) which causes blindness and cannot be reversed.[9] have performed random-forest classifier to classify the grade of DR which is focused not only in vessels segmentation of the retina but also skeletonization segmentations have been taken under consideration, the green channels have been extracted but the experiment been unsuccessful to recognize the severity of Diabetic Retinopathy through the identification of healthy and diabetic retinopathy patients is done.

The vision loss is something dangerous and this DR can be one of the reasons to cause vision loss. The expected range of diabetic retinopathy cases was 80 million by 2020 and the challenge was to detect the stage of DR but machine learning technique CNN was one of the optimal approaches to solving this. Since the Retinal data were in images and it was huge data CNN was consuming a lot of time.[10] In this TL(transfer learning) has been used to optimize the time consumed to train the data. This study was successful with an accuracy of 96.7% with 122 images of data in classifying the defected and normal retina.

DR is a common eye disease that leaks the blood from blood vessels of the retina and this blood fluids into the surrounding tissue which slowly affects the progress of vision. There are 2 types of DR (NPDR)non-proliferative diabetic retinopathy and (PDR) proliferative diabetic retinopathy. NPDR was sub derived into mild, moderate and severe. SVM (support vector machine) was implied to classify this severity range and executed with almost 95% accuracy which is sensible, and this work can be even more enhanced by applying texture analysis to get even better accuracy[11].

Diabetes mellitus results in DR by pathological changes of blood vessels of the retina. This DR cases often occurred in age between 20 -74 years. Over a few decades there are computed aided diagnosis(CAD) which is still helping the medical science the image diagnosis is also been the most practiced one. [12]With the help of images here in this research, DR has been classified in various classes by implying Bayesian statistical classifier to the images and achieved around 70% of accuracy. With the help of a Bayesian statistical classifier, Colors features have been used and classified each pixel into lesion or non-lesion classes in which accuracy of 100% has been achieved to identify retina [12].

Early Detection of DR is very important, huge investment on different techniques have been done as a requirement to detect DR in this experiment the detection of exudates have been sorted down to 3different stages: fetching exudate, extracting features, and ML(machine learning) techniques. In this experiment CNN(convolutional neural network) has been applied. With the data of images initially Optic Disc Detection has been removed and then Retinal Vessels have been removed from the images. Images were leftover with some dark lesions which have been also removed and CNN has been implied which has taken 91% accuracy on the test set. In this process, the network has been trained at around 150 epochs [13].

in [14] SVM has been used but, in this process green channel has been extracted from the images and taken it further with Curvelet transform which has boosted the accuracy around 5% as the performance of without Curvelet Transform islet with 86.32% of accuracy and whereas with Curvelet Transform has raised to 89.16%.

To achieve good accuracy using machine learning techniques [15] has elaborated on the preprocessing techniques by implementing greyscale on RGB and resizing the images in it which has given the accuracy of nearly 92% on CNN. Similarly, MLP (multilayer perception) has achieved good accuracy of 100% but [16] explained due to less volume of data the accuracy has been achieved to its maximum. In preprocessing the complex steps have been implied such as recoloring, feature extraction which has DCT (Discrete Cosine Transform), stat parameters, and formation of future optimization vector.

More information about eye disorders or diseases is detected with the help of CFI (color fundus image). For this, the data has been fetched based on CFI having a bright circular,

vascular over segments, highlights around blood vessels in retinas. With the help of 86 images in which half are normal and the remaining half are abnormal and performed SVM and ANN (artificial neural network). introduction of GLCM (Grey Level co-occurrence matrix) has been done to push the accuracy better with 87.5% from 84.6%.SVM has a better performance in classifying normal or DR of CFI [17].

[18]says DR can be detected by identifying the optic disk with maximum local variance. The optic disc was the only brighter part of retinal images and can be easily identified since the image works in RGB coordinates and with the help of 39 images, the MDD classifier is used to identify the Optic disc, and out of them, 35 abnormal images were detected. The identification of the Optic disc area is done and trimmed from the image.

DR is a disease that is caused by micro vascular retinal injury or changes of blood vessels inside the retina and may also lead to loss of vision.[2] The early detection of DR and providing it with the right treatment can be helpful. To identify the severity of DR the pre-processing of several images has been done in different stages. The detection of blood vessels and hemorrhages has been done and later removal of the Optical disc in the images has been performed and exudated them. The classification technique has been performed concerning features the accuracy of vessels and hemorrhages was resulted low at 80.5% comparatively whereas exudated images achieved 86.2% and there were few images unfocused in this process [19].

Over time the DR's risk of blindness has been reduced to 50%. Early diagnosis and treatment is the only way to prevent from the huge Damages.[4] Early identification techniques are computerized using laser and expensive tools. To give the best detection technique to identify DR Machine learning has achieved several ways successfully, here in this research Wavelet-based system is implied to detect exudates from retinal images. In this process more of cleaning images techniques used to achieve better accuracy compared to previous methods. RGB image has been converted to Greyscale, separation of the optic disc, exudates have been performed and density of exudated have been computed. The images have been classified into normal, mild, moderate, and severe levels. The accuracy of 95% has been achieved with the wavelet method which is decent and whereas previous methods achieved an accuracy of less than 92.7% and 94.7% using (Walter's and Ahmed's) approach [20].

III. METHODOLOGY

In this research Knowledge Discovery in Database (KDD) technique is used to classify the diabetic retinopathy disease. It is a research-based technique used to identify the hidden patterns in the database using data mining methods. Aim of the paper is to apply the neural network methods along with transfer learning to achieve the diabatic retinopathy disease classification through the multiple stages as mentioned in below figure. Data selection is the very first step and which is followed by the data preprocessing, data transformation, data mining technique and finally the results.

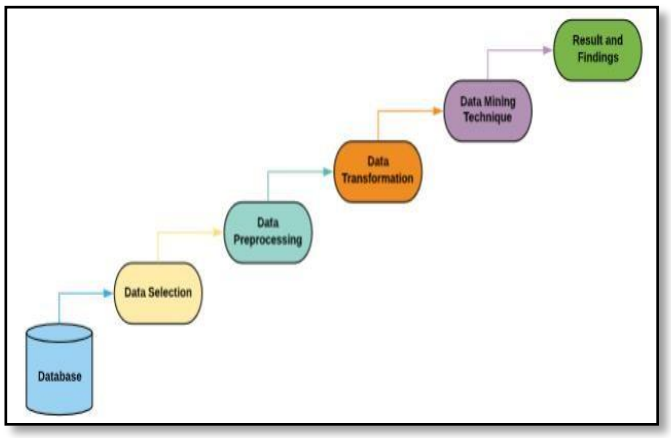

Figure 1: KDD for Diabetic Retinopathy Disease Classification

A. *Environemnt Setup*

Below table gives the brief idea about the environment setup required to perform the deep learning technique to identify the diabetic retinopathy disease.

Table 1: Environment Setup for Deep Learning Technique

Tool :	Google Colab
Specification :	12 GB RAM
Programming Language :	Python
Deep Learning Libraries :	TensorFlow and Keras

- Google Colab : It is a cloud-based service which offer to use the Graphical Processing Unit (GPU) and Tensor Processing Unit (TPU) for free to apply the machine learning methodology. It provides the numerous benefits like pre-installed environment setup with libraries, free GPU, large memory for storage and running application etc.
- Python Programming language: Python is a general-purpose language and object-oriented programming language. Python has large number of libraries and also frameworks to facilitate coding for Machine learning and Deep learning.
- TensorFlow: It is a open source library from Google which is data oriented ML library. It is used for numeric and Deep Neural Networks applications. TensorFlow consists of stable python API's.
- Keras: Keras is a high-level library that was built on TensorFlow where it provides API's for Neural Networks. Keras can run on GPU and CPU.

B. *Data Selection:*

Kaggle is the open sourced data library which contains numerous numbers of data for multiple domains. Diabetic retinopathy disease dataset is downloaded from the Kaggle which contain 35126 images of the eye. Images are labeled from 0 to 4 and severity of disease increase as the number increases. 0 means no disease and 4 means highest severity of the disease.

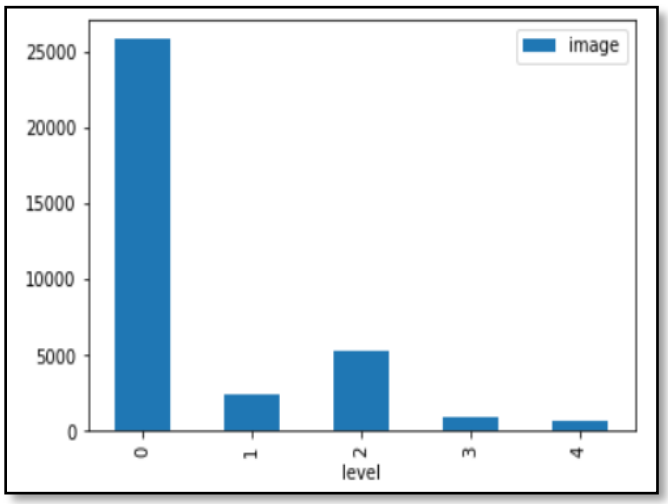

Figure 2: Image distribution based on different levels

Above figure shows that number of images for the label 0 is more and for other labels is quite low. Distribution of images improper and to use the equally distributed data it reduce the number of sample hence it will not produce the desire result.

C. Data Preprocessing:

Generally, data consist of noise which leads to inefficient information hence, data preprocessing is a vital step to remove the noisy data and generate the complete and consistent data. In image preprocessing it enhance the images for the better visibility and allow the machine to extract the correct features.

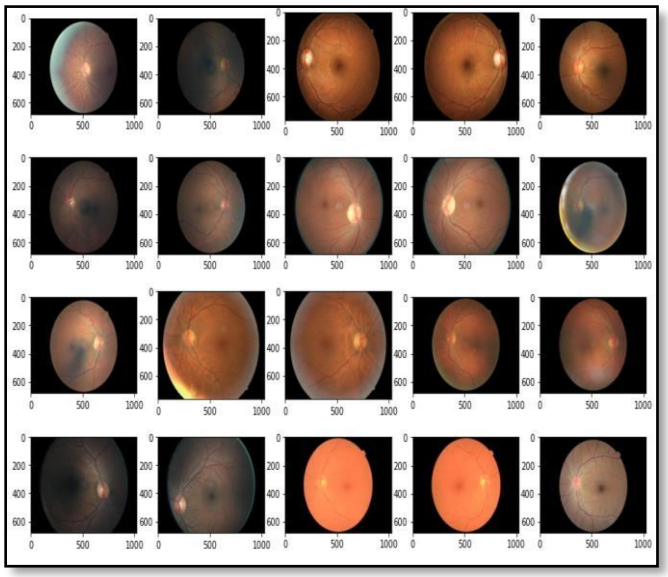

Figure 3a: Images Before Preprocessing

To improve the image number of steps has been applied, first the image is converted to RGB from BGR, image is resized to 320 X 320, gaussian filter is applied and finally weight is alpha blending is performed using addweighted function.

- **Gaussian Filter:** It is normally distributed, and function of probability have zero mean and one standard distribution [21]. It is a window and linear smoothing filter which contain the weighted mean to remove the noise from the image.
- **Gaussian Formula:**

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

Here x is the input parameter, sigma is standard deviation and sigma square is variance.

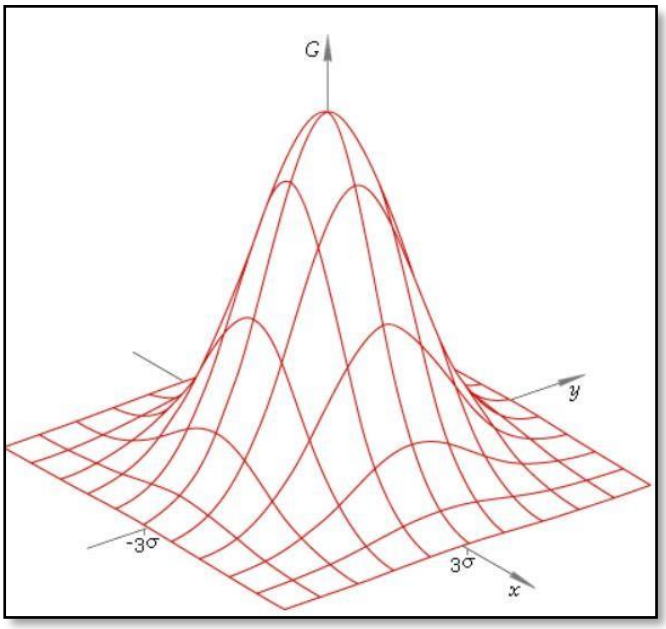

Figure 3b: Gaussian filter normally distributed

Below figure display the set of images after follow the above preprocessing step and it can be seen that now image is improved and each details can be seen with naked eye and will be helpful for the system to extract the exact features.

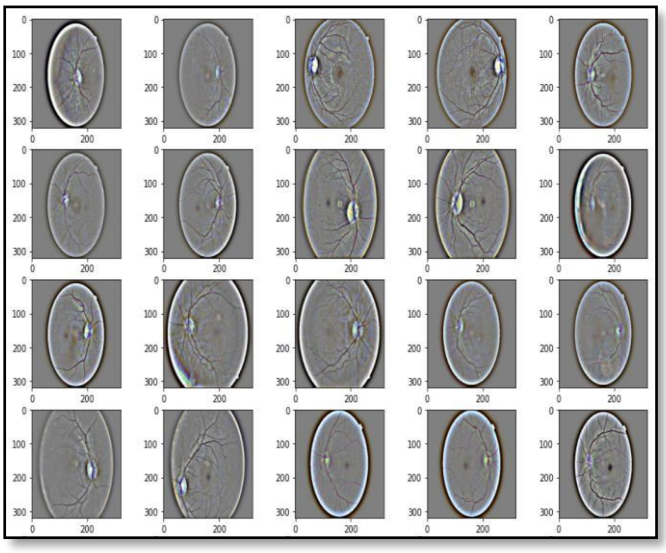

Figure 3c: Images After Preprocessing

D. Data Transformation:

Data transformation leads to organize the data in different structure to improve the quality of data which make easier for the computer system to understand. In out dataset there are 35126 images to train the model, but larger number of image

set can produce the positive impact on the accuracy of the model hence data augmentation is applied on the dataset.
Data sampling technique is used to divide the dataset in to training and testing sets. Here we have taken the 70:30 ratio so there are 24588 images for training and 10538 images for testing.
Data augmentation helps to increase the variety of image data to train the model. There are multiple data augmentation technique like rescaling, cropping, flipping, padding, rotation, brightness, contrast, saturation etc. It eventually helps to increase the data size and accuracy of the model.
In our paper we have implemented the following data augmentation technique using the keras ImageDataGenerator function:

- **Rescaling:** It helps to distribute the data effectively by multiplying or by dividing using some constant variable. Here, we have used the rescale parameter with constant 1/255.
- **Horizonal flip:** It flip the image along the horizonal (x) axis. It accepts the boolean value and by default its False, here we are using and set to True.
- **Vertical flip:** It flip the image along the vertical (y) axis. It also accepts the boolean value and by default its False, here we are using it also which is set to True.
- **Rotation range:** This parameter allows the image to rotate up to the provided range. In out model it is set to 360, means image can be rotate to any angle.
- **Zoom range:** We have applied the zoom range to 0.5, it means it can be zoom in or zoom out the image up to half range.

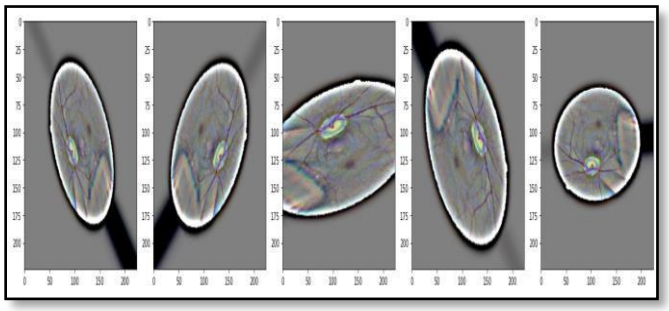

Figure 4: Image Augmentation train dataset

In above figure first image is rescaled and all other image are horizontally flipped, vertically flipped, rotated, zoom out respectively.

E. Data Mining Technique:

Convolutional Neural Network (CNN): Convolution Neural Networks are subset of Artificial Neural Networks. CNN take images as their input and assign weights and biases to different features in the images. CNN consists of layers that are in sequence to each layer. Multiple features are detected by feature detectors from the same image. Feature detector is small in size when compared to the input image and it will slide over the input image for detecting features all over the image.

Feature extraction: Feature extraction deals in reducing number of features in the dataset and create new features from the one that is existing. Transfer learning is based on the transfer of already learnt features to the other network in order to achieve better performance.

Hyperparameter optimization: It is used to optimize the neural network model by setting the parameters as per the model requirement. These parameters are tuned in a way that solve the deep learning model problem to reduce the loss function and improve the accuracy. There is different type of hyperparameter that can be used to optimize the model are mentioned in below table.

Table 2: Hyperparameter optimization

Hyperparameter	Value
1. Number of Layers	VGG use 19 CNN layers
2. Activation Function	ReLU, SoftMax
3. Batch size	32
4. Image size	224 X 224
5. Learning rate	0.000001
6. Optimizer	Adam
7. Loss Function	Categorical_crossentropy
8. Number of Epochs	10/30

- **Batch size:** Batch size is the number of training samples used in one iteration.
- **Learning rate:** Learning rate is one of the Hyperparameters that controls the changes in model with respect to the error. Based on error estimated every time, weights will update accordingly.
- **Optimizer:** In the training process, loss function and the model parameters were updated in accordance with output of loss function. This process of tuning the model to achieve fine accuracy is the work of optimizer. From the different optimizers, the one used for this dataset was Adam.
- **Loss Function:** Neural Networks try to minimize the error iteratively by updating their weights in network. This objective function is called Loss function.
- **Epochs:** Epoch is the measure of number of times that the dataset passed forward and backward in neural network. Updating weights in one pass is not sufficient for the model to learn, so the number of epochs is increased so as the weights are updated correspondingly.
- **Activation Function:** Activation function adds the weighted input with bias and then decide neurons to fire or not. It maps between 0 to 1 using ReLU.

This paper presents the Convolutional Neural Network using transfer learning are as below:

a. DenseNet121:

DenseNet stands for dense convolutional network which follow the feed forward neural network technique. It uses the less parameters to train the model which remove the feature redundancy. It contains minimum layers to produce the small set of features map which reduce the computational time. It concatenates the output feature of the previous layer to the input feature of next layer which means it add up the dimensions as input not the values as other pretrained model sums up the values. Dimensions of the feature do not change within the blocks, multiple filters are applied among them which is called transition layer and it utilize the batch

normalization and pooling layers. There are several advantages of densenet121, it reduces the gradient problem, utilize the feature reusability efficiently, utilize the smaller number of parameter and increase the feature extraction strength. Below figure display the architecture of denseNet121 which shows the flow of input feature from the different convolutional and transitional layers, it utilizes the average pooling and SoftMax activation function for the output.

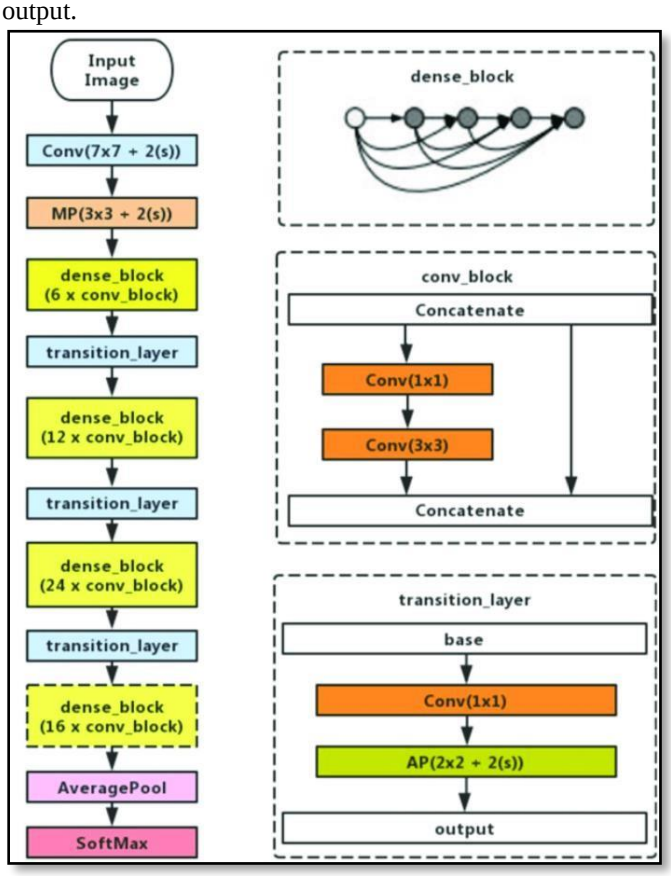

Figure 5: DenseNet121 Architecture

It can be seen in the below figures which explain the densenet121 summary model without fine-tuned model and with fine-tuned in figure 6a and 6b, respectively. In without fine tuning model weight of the pre-trained neural network remain constant it does not update to train the model whereas in fine tuning model weights gets updated to learn the features. Next, conv2d layers is used with ReLU activation function and the kernel with dimension 5 X 5 for the filtering followed by the drop out function which is used to drop the network layers for each epochs to vanish the overfitting of the model. Global average pooling is applied to compute the mean value for each feature map finally dense output layer is used with SoftMax activation function.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Functional)     (None, 7, 7, 1024)        7037504
_________________________________________________________________
conv2d (Conv2D)              (None, 5, 5, 32)          294944
_________________________________________________________________
global_average_pooling2d (Gl (None, 32)                0
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 5)                 165
=================================================================
Total params: 7,332,613
Trainable params: 295,109
Non-trainable params: 7,037,504
```

Figure 6a: Model Summary for DenseNet121 without Fine Tuning

Each figure shows the difference between number parameter used to train the model, above figure using less parameters to train as it is not training the densenet121 layer parameter and below figure train the parameter of pre-trained model as well.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Functional)     (None, 7, 7, 1024)        7037504
_________________________________________________________________
conv2d (Conv2D)              (None, 5, 5, 32)          294944
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 32)          0
_________________________________________________________________
global_average_pooling2d (Gl (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 5)                 165
=================================================================
Total params: 7,332,613
Trainable params: 7,248,965
Non-trainable params: 83,648
```

Figure 6b: Model Summary for DenseNet121 with Fine Tuning

b. VGG19:

VGG stands for the Visual Geometry Group which contains the 19 deep layers for the feature extraction. It contains the 16 Convolutional layers and 3 fully connected layers. Convolutional neural network uses this pre trained model to extract the features and it works very well with the small datasets [22]. This model utilizes the ImageNet dataset which contain the millions of images to train for the 1000 categories for the classification. By default, image input size for the VGG19 is 224 X 224.

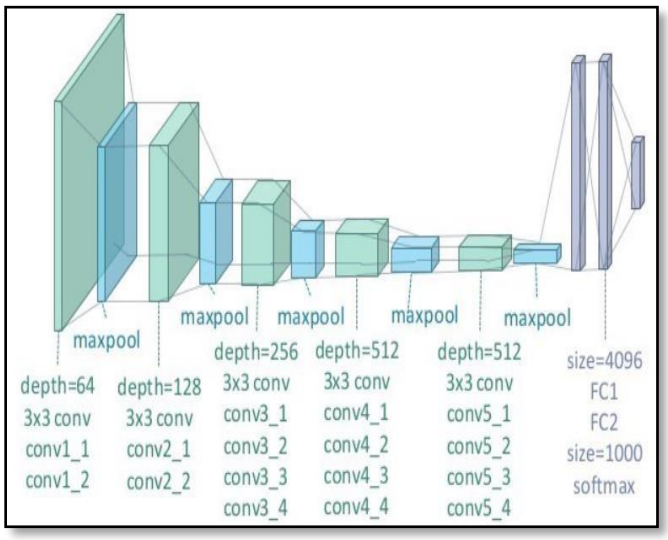

Figure 7: VGG19 Architecture

Above figure display the architecture of the VGG19. Depth 64 and 3X3 conv means square filters. Note that all the conv layers in VGG-19 use (3, 3) filters and that the number of filters increases in powers of two (64, 128, 256, 512). In all the Conv layers, stride length used in 1 pixel with a padding of 1 pixel as well on each side. There are 5 sets of conv layers, 2 of them have 64 filters, next set has 2 conv layers with 128 filters, next set has 4 conv layers with 256 filters, and next 2 sets have 4 conv layers each, with 512 filters. There are max pooling layers in between each set of conv layers. max pooling layers have 2x2 filters with stride of 2 pixels. The output of last pooling layer is flattened into a fully connected layer with 4096 neurons. The output goes to another fully connected layer with 4096 neurons, whose output is fed into another fully connected layer with 1000 neurons. All these layers are ReLU activated. Finally, there is a SoftMax layer which uses cross entropy loss.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 7, 7, 512)         20024384
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 5, 5, 32)          147488
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 32)          0
_________________________________________________________________
global_average_pooling2d_5 ( (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 165
=================================================================
Total params: 20,172,037
Trainable params: 147,653
Non-trainable params: 20,024,384
```

Figure 8a: Model Summary for VGG19 without Fine Tuning

Each figure shows the difference between number parameter used to train the model, above figure using less parameters to train as it is not training the densenet121 layer parameter and below figure train the parameter of pre-trained model as well.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 7, 7, 512)         20024384
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 5, 5, 32)          147488
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 32)          0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 165
=================================================================
Total params: 20,172,037
Trainable params: 20,172,037
Non-trainable params: 0
_________________________________________________________________
```

Figure 8b: Model Summary for VGG19 with Fine Tuning

IV. RESULT

A. DenseNet121:

```
Epoch 1/10
769/769 [==============================] - 430s 560ms/step - loss: 0.8123 - accuracy: 0.7346 - val_loss: 0.8109 - val_accuracy: 0.7364
Epoch 2/10
769/769 [==============================] - 423s 550ms/step - loss: 0.8042 - accuracy: 0.7352 - val_loss: 0.7869 - val_accuracy: 0.7368
Epoch 3/10
769/769 [==============================] - 425s 552ms/step - loss: 0.7989 - accuracy: 0.7364 - val_loss: 0.7848 - val_accuracy: 0.7361
Epoch 4/10
769/769 [==============================] - 425s 552ms/step - loss: 0.7934 - accuracy: 0.7371 - val_loss: 0.8043 - val_accuracy: 0.7359
Epoch 5/10
769/769 [==============================] - 425s 553ms/step - loss: 0.7942 - accuracy: 0.7371 - val_loss: 0.7897 - val_accuracy: 0.7374
Epoch 6/10
769/769 [==============================] - 431s 560ms/step - loss: 0.7889 - accuracy: 0.7373 - val_loss: 0.7848 - val_accuracy: 0.7358
Epoch 7/10
769/769 [==============================] - 448s 583ms/step - loss: 0.7849 - accuracy: 0.7381 - val_loss: 0.7776 - val_accuracy: 0.7380
Epoch 8/10
769/769 [==============================] - 449s 583ms/step - loss: 0.7850 - accuracy: 0.7387 - val_loss: 0.7781 - val_accuracy: 0.7374
Epoch 9/10
769/769 [==============================] - 452s 588ms/step - loss: 0.7858 - accuracy: 0.7375 - val_loss: 0.7796 - val_accuracy: 0.7362
Epoch 10/10
769/769 [==============================] - 455s 591ms/step - loss: 0.7835 - accuracy: 0.7379 - val_loss: 0.7917 - val_accuracy: 0.7369
```

Figure 9a: DenseNet Model Training without Fine Tuning

Above figure shows the DenseNet model trained on the 10 epochs without fine tuning and display accuracy of 73.69 % which remain almost same with increased number of epochs.

```
Epoch 1/10
769/769 [==============================] - 500s 651ms/step - loss: 0.8083 - accuracy: 0.7366 - val_loss: 0.7727 - val_accuracy: 0.7384
Epoch 2/10
769/769 [==============================] - 483s 628ms/step - loss: 0.7638 - accuracy: 0.7388 - val_loss: 0.7518 - val_accuracy: 0.7393
Epoch 3/10
769/769 [==============================] - 481s 625ms/step - loss: 0.7484 - accuracy: 0.7408 - val_loss: 0.7404 - val_accuracy: 0.7427
Epoch 4/10
769/769 [==============================] - 484s 629ms/step - loss: 0.7348 - accuracy: 0.7424 - val_loss: 0.7308 - val_accuracy: 0.7432
Epoch 5/10
769/769 [==============================] - 481s 625ms/step - loss: 0.7260 - accuracy: 0.7445 - val_loss: 0.7266 - val_accuracy: 0.7466
Epoch 6/10
769/769 [==============================] - 483s 628ms/step - loss: 0.7164 - accuracy: 0.7472 - val_loss: 0.7134 - val_accuracy: 0.7476
Epoch 7/10
769/769 [==============================] - 485s 631ms/step - loss: 0.7110 - accuracy: 0.7493 - val_loss: 0.7109 - val_accuracy: 0.7494
Epoch 8/10
769/769 [==============================] - 480s 624ms/step - loss: 0.7035 - accuracy: 0.7517 - val_loss: 0.7068 - val_accuracy: 0.7517
Epoch 9/10
769/769 [==============================] - 477s 620ms/step - loss: 0.6954 - accuracy: 0.7531 - val_loss: 0.7027 - val_accuracy: 0.7519
Epoch 10/10
769/769 [==============================] - 478s 622ms/step - loss: 0.6851 - accuracy: 0.7584 - val_loss: 0.7030 - val_accuracy: 0.7583
```

Figure 9b: DenseNet Model Training with Fine Tuning (10 Epochs)

DenseNet layer is used with fine tuning as the accuracy of the model was same for all the epochs and in above figure accuracy of the model is increased to 75.83 %. As the number of epochs affect the accuracy below figure display the accuracy for 30 epochs and it increased to 78.74 %, hence it concludes that number of increments in epochs increase the accuracy of the model using fine tuning.

```
Epoch 14/30
769/769 [==============================] - 516s 671ms/step - loss: 0.6356 - accuracy: 0.7837 - val_loss: 0.6572 - val_accuracy: 0.7785
Epoch 15/30
769/769 [==============================] - 522s 679ms/step - loss: 0.6273 - accuracy: 0.7870 - val_loss: 0.6602 - val_accuracy: 0.7813
Epoch 16/30
769/769 [==============================] - 523s 679ms/step - loss: 0.6231 - accuracy: 0.7900 - val_loss: 0.6511 - val_accuracy: 0.7829
Epoch 17/30
769/769 [==============================] - 526s 684ms/step - loss: 0.6189 - accuracy: 0.7901 - val_loss: 0.6691 - val_accuracy: 0.7860
Epoch 18/30
769/769 [==============================] - 531s 690ms/step - loss: 0.6125 - accuracy: 0.7919 - val_loss: 0.6536 - val_accuracy: 0.7852
Epoch 19/30
769/769 [==============================] - 532s 692ms/step - loss: 0.6030 - accuracy: 0.7945 - val_loss: 0.6640 - val_accuracy: 0.7826
Epoch 20/30
769/769 [==============================] - 534s 695ms/step - loss: 0.6013 - accuracy: 0.7953 - val_loss: 0.6498 - val_accuracy: 0.7855
Epoch 21/30
769/769 [==============================] - 537s 698ms/step - loss: 0.5958 - accuracy: 0.7939 - val_loss: 0.6492 - val_accuracy: 0.7850
Epoch 22/30
769/769 [==============================] - 535s 696ms/step - loss: 0.5947 - accuracy: 0.7968 - val_loss: 0.6629 - val_accuracy: 0.7841
Epoch 23/30
769/769 [==============================] - 541s 703ms/step - loss: 0.5865 - accuracy: 0.7998 - val_loss: 0.6506 - val_accuracy: 0.7872
Epoch 24/30
769/769 [==============================] - 542s 705ms/step - loss: 0.5868 - accuracy: 0.7979 - val_loss: 0.6547 - val_accuracy: 0.7838
Epoch 25/30
769/769 [==============================] - 542s 704ms/step - loss: 0.5841 - accuracy: 0.7994 - val_loss: 0.6544 - val_accuracy: 0.7896
Epoch 26/30
769/769 [==============================] - 538s 700ms/step - loss: 0.5769 - accuracy: 0.8031 - val_loss: 0.6593 - val_accuracy: 0.7879
Epoch 27/30
769/769 [==============================] - 537s 698ms/step - loss: 0.5755 - accuracy: 0.8032 - val_loss: 0.6465 - val_accuracy: 0.7892
Epoch 28/30
769/769 [==============================] - 535s 695ms/step - loss: 0.5716 - accuracy: 0.8022 - val_loss: 0.6493 - val_accuracy: 0.7894
Epoch 29/30
769/769 [==============================] - 535s 696ms/step - loss: 0.5668 - accuracy: 0.8058 - val_loss: 0.6432 - val_accuracy: 0.7877
Epoch 30/30
769/769 [==============================] - 513s 667ms/step - loss: 0.5663 - accuracy: 0.8059 - val_loss: 0.6489 - val_accuracy: 0.7874
```

Figure 9c: DenseNet Model Training with Fine Tuning (30 Epochs)

Below figure display the graphical representation of accuracy and loss for the 30 epochs using fine tuning for DenseNet121 model.

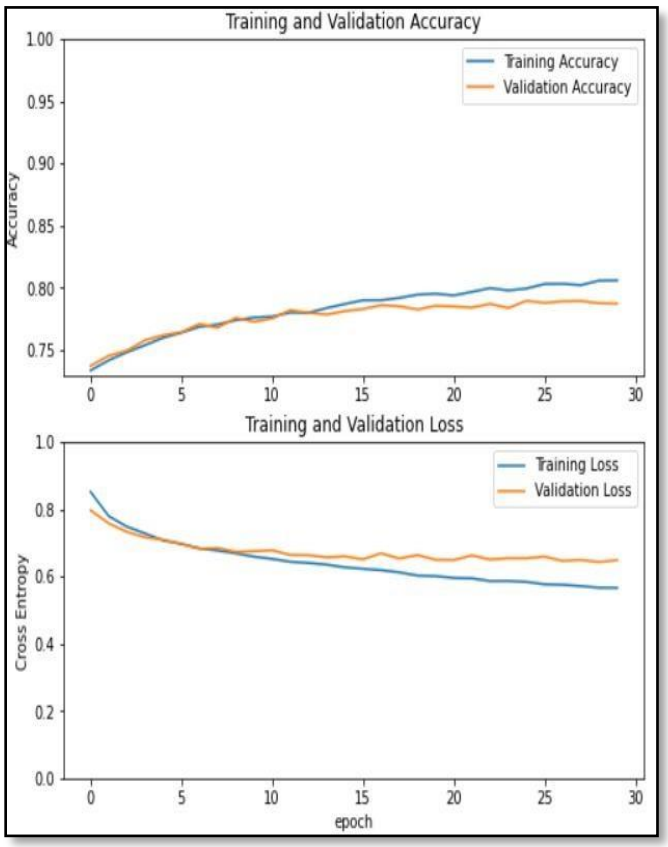

Figure 10: DenseNet Model Accuracy Vs Loss

B. VGG19:

```
Epoch 1/10
769/769 [==============================] - 434s 565ms/step - loss: 0.8708 - accuracy: 0.7336 - val_loss: 0.8344 - val_accuracy: 0.7350
Epoch 2/10
769/769 [==============================] - 436s 567ms/step - loss: 0.8487 - accuracy: 0.7346 - val_loss: 0.8410 - val_accuracy: 0.7350
Epoch 3/10
769/769 [==============================] - 436s 567ms/step - loss: 0.8403 - accuracy: 0.7347 - val_loss: 0.8312 - val_accuracy: 0.7348
Epoch 4/10
769/769 [==============================] - 438s 570ms/step - loss: 0.8356 - accuracy: 0.7347 - val_loss: 0.8253 - val_accuracy: 0.7349
Epoch 5/10
769/769 [==============================] - 438s 570ms/step - loss: 0.8314 - accuracy: 0.7346 - val_loss: 0.8244 - val_accuracy: 0.7350
Epoch 6/10
769/769 [==============================] - 438s 570ms/step - loss: 0.8327 - accuracy: 0.7347 - val_loss: 0.8224 - val_accuracy: 0.7347
Epoch 7/10
769/769 [==============================] - 435s 565ms/step - loss: 0.8302 - accuracy: 0.7346 - val_loss: 0.8257 - val_accuracy: 0.7350
Epoch 8/10
769/769 [==============================] - 441s 574ms/step - loss: 0.8268 - accuracy: 0.7347 - val_loss: 0.8172 - val_accuracy: 0.7349
Epoch 9/10
769/769 [==============================] - 439s 571ms/step - loss: 0.8269 - accuracy: 0.7348 - val_loss: 0.8218 - val_accuracy: 0.7347
Epoch 10/10
769/769 [==============================] - 439s 571ms/step - loss: 0.8258 - accuracy: 0.7349 - val_loss: 0.8258 - val_accuracy: 0.7350
```

Figure 11a: VGG Model Training without Fine Tuning

Above figure shows the VGG19 model trained on the 10 epochs without fine tuning and display accuracy of 73.50 % which remain almost same with increased number of epochs.

```
Epoch 1/10
  2/769 [..............................] - ETA: 1:38 - loss: 1.1115 - accuracy: 0.7188WARNING:tensorflow:Callbacks method `on_train_bat
769/769 [==============================] - 503s 654ms/step - loss: 0.8061 - accuracy: 0.7347 - val_loss: 0.7631 - val_accuracy: 0.7351
Epoch 2/10
769/769 [==============================] - 501s 651ms/step - loss: 0.7647 - accuracy: 0.7350 - val_loss: 0.7510 - val_accuracy: 0.7350
Epoch 3/10
769/769 [==============================] - 494s 642ms/step - loss: 0.7506 - accuracy: 0.7359 - val_loss: 0.7462 - val_accuracy: 0.7354
Epoch 4/10
769/769 [==============================] - 485s 630ms/step - loss: 0.7400 - accuracy: 0.7367 - val_loss: 0.7253 - val_accuracy: 0.7388
Epoch 5/10
769/769 [==============================] - 480s 624ms/step - loss: 0.7297 - accuracy: 0.7373 - val_loss: 0.7290 - val_accuracy: 0.7377
Epoch 6/10
769/769 [==============================] - 479s 623ms/step - loss: 0.7252 - accuracy: 0.7382 - val_loss: 0.7180 - val_accuracy: 0.7382
Epoch 7/10
769/769 [==============================] - 477s 620ms/step - loss: 0.7173 - accuracy: 0.7394 - val_loss: 0.7139 - val_accuracy: 0.7382
Epoch 8/10
769/769 [==============================] - 478s 622ms/step - loss: 0.7126 - accuracy: 0.7406 - val_loss: 0.7054 - val_accuracy: 0.7396
Epoch 9/10
769/769 [==============================] - 477s 620ms/step - loss: 0.7067 - accuracy: 0.7439 - val_loss: 0.7215 - val_accuracy: 0.7411
Epoch 10/10
769/769 [==============================] - 476s 619ms/step - loss: 0.6992 - accuracy: 0.7495 - val_loss: 0.7010 - val_accuracy: 0.7551
```

Figure 11b: VGG Model Training with Fine Tuning (10 Epochs)

VGG19 layer is used with fine tuning as the accuracy of the model was same for all the epochs and in above figure accuracy of the model is increased to 75.51 %. As the number of epochs affect the accuracy below figure display the accuracy for 30 epochs and it increased to 79 %, hence it concludes that number of increments in epochs increase the accuracy of the model using fine tuning.

```
769/769 [==============================] - 894s 1s/step - loss: 0.6581 - accuracy: 0.7758 - val_loss: 0.6713 - val_accuracy: 0.7772
Epoch 14/30
769/769 [==============================] - 893s 1s/step - loss: 0.6544 - accuracy: 0.7796 - val_loss: 0.6618 - val_accuracy: 0.7753
Epoch 15/30
769/769 [==============================] - 904s 1s/step - loss: 0.6464 - accuracy: 0.7828 - val_loss: 0.6564 - val_accuracy: 0.7837
Epoch 16/30
769/769 [==============================] - 904s 1s/step - loss: 0.6476 - accuracy: 0.7793 - val_loss: 0.6561 - val_accuracy: 0.7806
Epoch 17/30
769/769 [==============================] - 894s 1s/step - loss: 0.6419 - accuracy: 0.7824 - val_loss: 0.6742 - val_accuracy: 0.7744
Epoch 18/30
769/769 [==============================] - 896s 1s/step - loss: 0.6352 - accuracy: 0.7838 - val_loss: 0.6623 - val_accuracy: 0.7822
Epoch 19/30
769/769 [==============================] - 899s 1s/step - loss: 0.6331 - accuracy: 0.7846 - val_loss: 0.6449 - val_accuracy: 0.7865
Epoch 20/30
769/769 [==============================] - 897s 1s/step - loss: 0.6264 - accuracy: 0.7871 - val_loss: 0.6550 - val_accuracy: 0.7808
Epoch 21/30
769/769 [==============================] - 895s 1s/step - loss: 0.6266 - accuracy: 0.7879 - val_loss: 0.6622 - val_accuracy: 0.7794
Epoch 22/30
769/769 [==============================] - 901s 1s/step - loss: 0.6222 - accuracy: 0.7896 - val_loss: 0.6598 - val_accuracy: 0.7838
Epoch 23/30
769/769 [==============================] - 898s 1s/step - loss: 0.6203 - accuracy: 0.7907 - val_loss: 0.6532 - val_accuracy: 0.7844
Epoch 24/30
769/769 [==============================] - 896s 1s/step - loss: 0.6167 - accuracy: 0.7919 - val_loss: 0.6597 - val_accuracy: 0.7794
Epoch 25/30
769/769 [==============================] - 894s 1s/step - loss: 0.6107 - accuracy: 0.7924 - val_loss: 0.6509 - val_accuracy: 0.7891
Epoch 26/30
769/769 [==============================] - 898s 1s/step - loss: 0.6070 - accuracy: 0.7947 - val_loss: 0.6510 - val_accuracy: 0.7818
Epoch 27/30
769/769 [==============================] - 894s 1s/step - loss: 0.6073 - accuracy: 0.7955 - val_loss: 0.6361 - val_accuracy: 0.7851
Epoch 28/30
769/769 [==============================] - 894s 1s/step - loss: 0.6025 - accuracy: 0.7935 - val_loss: 0.6391 - val_accuracy: 0.7896
Epoch 29/30
769/769 [==============================] - 894s 1s/step - loss: 0.6024 - accuracy: 0.7952 - val_loss: 0.6476 - val_accuracy: 0.7852
Epoch 30/30
769/769 [==============================] - 896s 1s/step - loss: 0.6005 - accuracy: 0.7944 - val_loss: 0.6363 - val_accuracy: 0.7900
```

Figure 11c: VGG Model Training with Fine Tuning (30 Epochs)

Below figure display the graphical representation of accuracy and loss for the 30 epochs using fine tuning for DenseNet121 model.

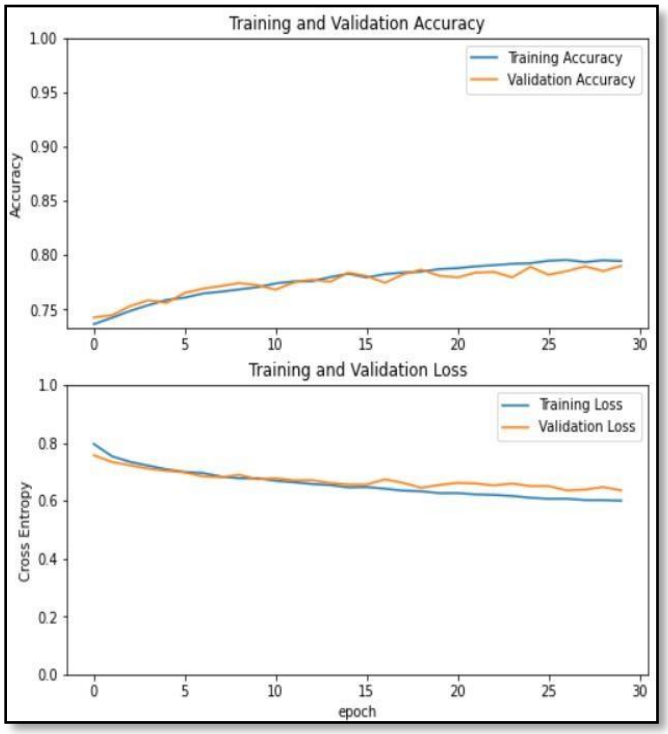

Figure 12: VGG Model Accuracy Vs Loss

C. Evaluation Summary:

Table 3: Accuracy Comparison of DenseNet VS VGG

Models / Measures	DenseNet121	VGG19
Accuracy without Fine tuning (10 Epochs)	73.69 %	73.50 %
Accuracy with Fine tuning (10 Epochs)	75.83 %	75.51 %
Accuracy with Fine tuning (30 Epochs)	**78.74 %**	**79.00 %**

V. CONCLUSION AND FUTURE WORK

Detecting the severity of Diabetic retinopathy has been achieved successfully, Deep learning techniques have been used, the data set is tested using two different pre-trained models VGG19 and DenseNet on convolutional neural network. Image processing has been performed using gaussian filter technique to remove the noise and image enhancement. To increase the trained sample data augmentation technique is applied by rescale the image, flipping technique, rotation etc. Hyperparameter optimization is used to select the best set of parameters to achieve the higher accuracy. Convolution neural network is applied using transfer learning i.e DenseNet121 and VGG19. CNN model used the kernel size of 5 X 5, ReLU activation function to produce the linear data, dropout technique is used to reduce the overfitting of the model, categorical cross entropy loss function is used as dataset contains the multi class label finally, ADAM optimizer is used with lower learning rate for the better accuracy.

It has been noticed that increasing the number of epochs produce the higher accuracy for the model. For 30 epochs accuracy for the DenseNet121 and VGG19 model is 78.74 % and 79 % respectively which means VGG performed better than DenseNet for the classification of diabetic retinopathy severity.

In future the task to identify the DR(Diabetic Retinopathy) can be improved by adding more pre-processing techniques like ISR (Image Super Resolution), modification in hyperparameters optimization also leads to improve in accuracy.

REFERENCES

[1] R. Ramani, J. Shanthamalar J. and B. Lakshmi, “Automatic Diabetic Retinopathy Detection Through Ensemble Classification Techniques Automated Diabetic Retionapthy Classification,” 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Coimbatore, 2017.

[2] H. Jiang, K. Yang, D. Zhang, M. Gao, H. Ma and W. Qian, “An Interpretable Ensemble Deep Learning Model for Diabetic Retinopathy Disease Classification,” 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, 2019.

[3] M. Pandey and M. Arora “Deep Neural Network for Diabetic Retinopathy Detection,” 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019.

[4] S. Wang et al. “Diabetic Retinopathy Diagnosis Using Multichannel Generative Adversarial Network With Semisupervision,” in IEEE Transactions on Automation Science and Engineering, 2020.

[5] N. Harun, F.Hassan, Y.Yusof, and Z. Embong, “Classification of Fundus Images For Diabetic Retinopathy using Artificial Neural Network,” IEEE, 2019.

[6] M. Jena, D. Mishra and S. Mishra, “Detection of Diabetic Retinopathy Images Using a Fully Convolutional Neural Network,” 2018 2nd International Conference on Data Science and Business Analytics (ICDSBA), 2018.

[7] D.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), 2019.

[8] S. Kanth, M. Kakkar and A. Jaiswal, “Identification of different stages of Diabetic Retinopathy using artificial neural network,” 2013 Sixth International Conference on Contemporary Computing (IC3), Noida, 2013.

[9] F. Alzami, Abdussalam, R. A. Megantara, A. Z. Fanani and Purwanto, "Diabetic Retinopathy Grade Classification based on Fractal Analysis and Random Forest," 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), Semarang, Indonesia, 2019.

[10] S. Gupta, S. Goel, A. Panwar ,A. Mittal, R. Nijhawan and A. K. Singh, "Classification of Lesions in Retinal Fundus Images for Diabetic Retinopathy Using Transfer Learning," 2019 International Conference on Information Technology (ICIT), Bhubaneswar, India, 2019.

[11] E. V. Carrera, R. Carrera and A. González, “Automated detection of diabetic retinopathy using SVM,” 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Cusco, 2017.

[12] R. Maher, S. Kayte, and D. M. Dhopeshwarkar, “Review of automated detection for diabetes retinopathy using fundus images,” International Journal of Advanced Research in Computer Science and Software Engineering, 2015.

[13] S. Yu, D. Xiao and Y. Kanagasingam, "Exudate detection for diabetic retinopathy with convolutional neural networks," 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Seogwipo, 2017.

[14] D. Selvathi and N. Balagopal, “Detection of retinal blood vessels using curvelet transform,” 2012 International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, 2012.

[15] N. Chakrabarty, "A Deep Learning Method for the detection of Diabetic Retinopathy,” 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gorakhpur, 2018, pp. 1-5

[16] G. U. Kharat and A. P. Bhatkar, “Detection of Diabetic Retinopathy in Retinal Images Using MLP Classifier," 2015 IEEE International Symposium on Nanoelectronic and Information Systems, Indore, 2015.

[17] G. A. Padmanabha, M. A. Appaji, M. Prasad, S. Joshi and H. Lu, "Classification of diabetic retinopathy using textural features in retinal color fundus image,”2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, 2017.

[18] S. N. Narayanan and V. V. Kumari “Diabetic Retinopathy-Early Detection Using Image Processing Techniques”, International Journal on Computer Science and Engineering.

[19] Z. A. Omar, M. Hanafi, S. Mashohor, N. F. M. Mahfudz and M. Muna'im, “Automatic diabetic retinopathy detection and classification system," 2017 7th IEEE International Conference on System Engineering and Technology (ICSET), Shah Alam, 2017

[20] A. S. Jadhav and P. B. Patil, “Detection of exudates for diabetic retinopathy using wavelet transform," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, 2017.

[21] F. Talbi, F. Alim, S. Seddiki, I. Mezzah and B. Hachemi, “Separable convolution gaussian smoothing filters on a xilinx FPGA platform”, Fifth International Conference on the Innovative Computing Technology (INTECH 2015), Pontevedra, Spain, 20-22 May 2015.

[22] Long Wen, X. Li, Xinyu Li and Liang Gao, “A New Transfer Learning Based on VGG-19 Network for Fault Diagnosis”, 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD), Porto, Portugal, Portugal, 6-8 May 2019.