ABSTRACT

Patients suffering from prolonged diabetic conditions are prone to Diabetic Retinopathy (DR) which leads to vision impairment if left untreated. Diabetic Retinopathy has been on the rise across the globe due to an increase in the number of diabetic patients. Diabetic Retinopathy detection in early stages has become vital to prevent permanent vision impairment and avoid arduous medical treatment in the later stages. Diabetic Retinopathy (DR) causes damage to the retina and gradual loss of sight and in severe cases permanent vision impairment eventually leading to blindness. An early analysis of Diabetic Retinopathy helps in controlling the progress of the disease and increases the chances of recovery. An automated classification of Diabetic Retinopathy using images is a difficult job due to the microscopic variability of the appearance of different classes and the lack of a standard data infrastructure by medical professionals. One of the major deterrents in automated Diabetic Retinopathy (DR) detection is the identification of the essential features in the fundus image. Techniques like Gaussian Blur and auto-cropping has been used for feature extraction and noise removal. Through this research, we aim to classify various fundus images of the eye into various classes of diabetic Retinopathy and automate the screening process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Network
DR	Diabetic Retinopathy
SGD	Stochastic gradient descent
ACC	Accuracy
AUC	Area Under Curve
SVM	Support Vextor Machine
FUNC	function

LIST OF SYMBOLS

η	Learning Rate
θ	Vector of parameters

CHAPTER 1

INTRODUCTION

1.1 Diabetic Retinopathy

Over the years there has been a drastic increase in the number of people affected by diabetes. Diabetes is complemented by various other diabetic conditions which generally affect people who have had extended suffering from diabetes. It is the major leading cause of blindness in this present day and age. It has affected almost 93 million people in the world. It causes gradual loss of sight which leads to blindness in the long run. All kinds of diabetic patients are affected by DR in due course. DR causes the blood vessels present in the eye to leak blood and other fluids around the retina. As the severity of DR increases, it causes the blood vessels to get blocked hence the retina stops receiving proper nourishment. The retina starts to become unhealthy which eventually leads to loss of sight.

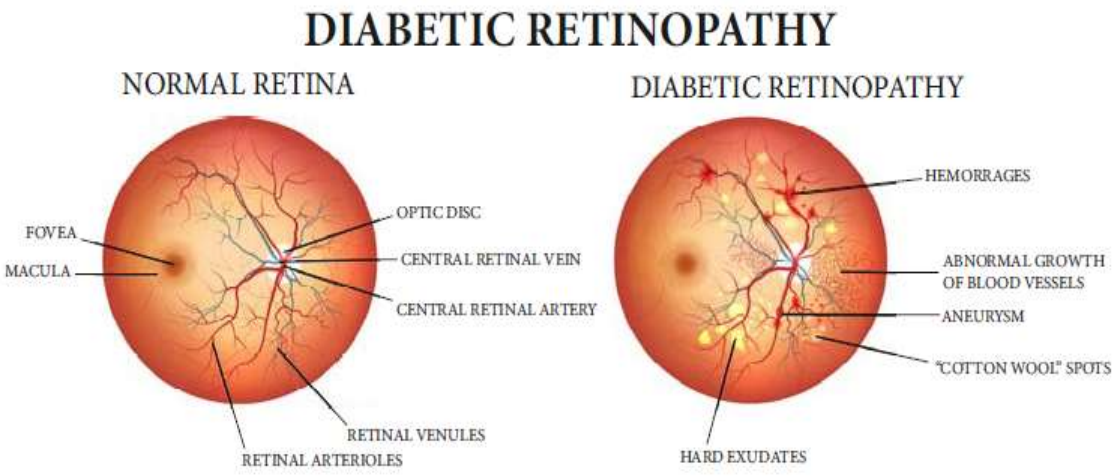


Figure 1.1: Normal v/s Diabetic Retinopathy

1.2 Need for an Automated System

Early diagnosis is of the essence in the treatment of Diabetic Retinopathy. If we can analyze the Diabetic Retinopathy early, then there is a great possibility of containing the disease and hence avoiding further escalation. Conventional methods to detect diabetic retinopathy require regular screening of the eye. In the current scenario, there is a lot of expert manpower required for the present screening facilities that are needed for this purpose. Therefore, there is a need to place an automated screening technique in its place. By doing this it will drastically reduce the workload and give quicker outcomes and hence plays a helping hand to the ophthalmologists.

With advancements in machine learning, various algorithms are being used to generate an automated system to carry out the screening process. However, in recent times, various techniques such as Convolutional Neural Network (CNN), transfer learning have brought in drastic changes in the field of medical science. Researchers have started utilizing deep learning as a method to perform image segmentation and classification of Diabetic Retinopathy.

1.3 Deep Learning

Deep learning is a method that emulates the functioning of the human brain which gathers all the data and creates a knowledge base. Using this knowledge it extracts various information like patterns and takes decisions on it. The deep learning model is first trained on a certain set of labeled and structured databases and then tested onto the other set of databases. Hence, deep learning helps in deploying a computer model that helps in classifying data into different categories.

Deep learning model has proven to have high computational powers as well as has been able to extract a multitude of features from the given labeled data set. Hence various deep learning methods like convolutional neural networks are being used in various medical institutes, . Hence deep learning algorithms have had a greater impact in today's classification and detection model of image processing, text processing etc.

CHAPTER 2

LITERATURE SURVEY

The current methodology of research employs various machine learning algorithms for the diagnosis of diseases along with the help of available medical records for the classification and detection of various diseases. An intensive survey was carried out to gain knowledge on the methods applied by researchers and to expand our knowledge of various pre-processing methods. This survey also helped us know about various data augmentation procedures that could help in getting more generalized data sets.

The approach used by Ankita Gupta and Rita Chhikarab [1] divides their detection of DR into two different approaches. The first approach focuses on Blood Vessel Segmentation and the second on the Identification of lesions. In the first approach the primary aim is to segregate the blood vessels from the background. this process not only helps in detecting Diabetic Retinopathy (DR) but also helps in detecting various other diseases such as other ophthalmic diseases and cardiovascular diseases. Detecting Micro aneurysms is one of the key fundamental steps done when following the second approach. The paper also indicates the importance of enhancing the images, therefore making the bad images a viable data in training the model. This paper helps us in understanding and gaining knowledge about various results like sensitivity, Area Under Curve (AUC), the Accuracy (ACC) that was derived when different machine learning algorithms were used such as Improved Match Filtering, Ensemble Classifier, etc.

The algorithm applied by Yuchen Wu and Ze Hu [2] uses the Migration Learning Approach which is one of the new machine learning approaches. There are four different approaches to implement this method. They have used Feature-based Transfer Learning. The paper talks about various data enhancement techniques due to the imbalanced data set. One of the fundamental enhancement techniques is the predifined image data generator made available by keras. The pre-training model they used is based on

ImageNet. Since there was a requirement of data augmentation which involved the usage of ImageDataGenerator which is a class present in Keras. Hence Keras was primarily used. They used the pre-training model for feature extraction that was required to develop the final model to detect DR.

The results obtained by using CNN has been projected by the paper published by Shuang Yu, Di Xiao and Yogesan Kanagasingam [3] It uses pixel-wise exudate image patches. The accuracy that was obtained was off 91.92%. The Framework for exudate detection with deep learning includes three main procedures before the image is sent to the DR model that was established using CNN. The three major steps are Removal of Optic Disc Detection, Removal of Retinal Vessels, and Ultimate Opening. Since the exudates and the Optic disc are bright in nature hence masking out optic disc is a crucial step.This step was carried out with the help of Local Phase Summetry. The images are entered in the form of 64*64 patches. This paper uses Convolutional Neural Network due its previous success in the field of image classification and recognition. In CNN the features are learned automatically unlike that of SVM. It also indicates the importance of add a pooling layer alternatively to the convolutional so as to reduce the spatial dimension.

There have been various modern screening approaches that have been used to diagnose Diabetic Retinopathy. S.D. Shirbahadurkar, V. M. Mane and D. V. Jadhav[4] have tried to use decision trees to diagnose DR. During the pre-processing stage the images are converted into gray scale images using the gray scale function. The gray scale images help reduce the computational power, then these images are converted to binary images . This process helps in blood vessel segmentation. This algorithm can classify the fundus image into various DR. Hence this algorithm is holoentropy enabled. The above method uses activities like grey scaling (converts image into a pixel driven image that a computer understands), optic disc segmentation and blood vessel segmentation to detect DR. The following steps include feature extraction and hence a feature extraction vector to which appropriate weights are assigned to different features which help in detecting DR.

There also have been various approaches for detecting proliferative DR. This re-

search was published by Anaswara Chandran, Prof. Nisha K K and Dr. Vineetha S [5]. The ability to handle higher dimensional feature sets was essential for the classification process. Hence, Random-Forest Classifier was used here.The data set used here was of MESSIDOR and STARE. There were three main categorical classes in which the dataset was divided namely-No DR,NPDR and PDR. Different patches of the data set are used to train various decision trees. The patches are extracted from the image sets. The patches undergo two distinct extraction processes namely Texture Feature Extraction and Vessel Feature Extraction.The Texture extraction consists of four main sub-processes. AM-FM Characterization method uses Amplitude modulation and frequency modulation to detect numerous textures present in the image. Secondly, Log Gabor Responses . This process is used for edge detection of the image. The main advantage that the Random Forest Classifier provides is the ability to compute feature sets of higher dimensions. The random forest uses the different outputs attained by different trees . The decision trees after giving various decisions decisions as its output is fed forward to random forest classifier which uses a rule-based approach to classify the images to reach a certain result.

The research methodology employed by Mamta Arora and Mrinal Pandey of Manav Rachna University [6] shows the various stages involved in using deep neural networks for the diagnosis of diabetic retinopathy detection. This tries to classify the images to 5 different classes of DR. The images are of varied sizes hence these are resized to a fixed size of 128 * 128, this is done before feeding it to the training model. They followed a two-step process which includes step 1: data preprocessing and augmentation and step2: convolution layer. To establish uniformity across all the categorical classes we make use of data generator which is the integral part of data augmentation . The convolutional layer was further divided into a 5 step process that involved the convolutional layer which is the basic building block of Convolutional neural extracting weights, Pooling Layer, connecting layer and logistic classifier. The basic building block of Convolutional neural extracting weights has numerous neurons which help in calculating weights of the attribute. The Pooling layer provides three distinct pooling layers namely- min pooling, max pooling and average pooling. It is a layer in which all the previous layer neuron is connected to every neuron in the next layer.

Placing emphasis on non-proliferative DR, Support Vextor Machine (SVM) algorithm was used for the diagnoses. This is shown by Handayani Tjandrasa Ricky Eka Putra, Arya Yudhi Wijaya and Isye Arieshanti[7] and Enrique V. Carrera, Andres Gonzalez and Ricardo Carrera[8] . [8]. The main features of this paper were exudate segmentation, feature extraction and the classification of NPDR severity level. The extracted features were finally trained and tested using soft margin SVM as a classification model.

According to Handayani Tjandrasa, Ricky Eka Putra, Arya Yudhi Wijaya and Isye Arieshanti[7], Support Vector Machine helps in finding the optimum hyperplane. This hyperplane helps in classifying two distinct categorizes with largest difference. The paper carries out three systematic process for detecting diabetic retinopathy. The first process is called exudate segmentation, this process is carried out by morphology method. The features are then extracted from the segmented image. Then the SVM classifies the images into various categories. Before feeding the images to the training model the image are resize to a uniform size of 576 * 720. The grays scaling then helps to gray scale the images. The images are then enhanced by using morphology method. Similarly, the optic disc present in the images are masked out . To extract features, the exudated image is fed to the training model. the model then creates a feature vector set used for classifying the image. Therefore SVM is able to classify the image.

Enrique V. Carrera, Andres Gonzalez and Ricardo Carrera[8] also uses SVM algorithm to classify the image. The process of extracting features staers off with determining the density of blood vessels. One of the most vital features that is extracted are the micro aneurysems. The fundus image of the eye consists of small lump like structures in blood vessels. The number of micro aneurysems play a key role in recognising the class of the image. The following are the features that help in classifying-Standard deviation of the red component.

- Standard deviation of the red component.
- Standard deviation of the green component.
- Standard deviation of the blue component.

- Blood vessel density.
- Possible number of microaneurysms.
- Actual number of microaneurysms.
- Density of hard exudates.
- Green component entropy.

In the proposed system we have used Convolutional Neural Network and ResNet50. Convolutional Neural Network is one the more popular algorithm used in image classification. The literature Survey also indicates towards the effectiveness of Convolutional Neural Network.The need for good pre-processing is also inferred from the survey conducted. Hence we have chosen various preprocessing steps to enhance the data. Data Augmentation is also a very vital process in balancing and making the data uniform.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Dataset

The dataset consists of fundus images of the eye which were obtained through various medical professionals. For the development of dataset, macula-centered fundus images were obtained from EYEPACS which is a U.S. based private organization and three other hospitals in India. The dataset consisted of fundus images from various patients with varying levels of illumination and other physical parameters leading to inconsistencies with the data even in the same class. The dataset consists of three categorical classes of Diabetic Retinopathy. The classes being No DR, Non-proliferative DR, proliferative DR.

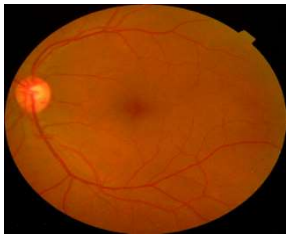


Figure 3.1: No DR

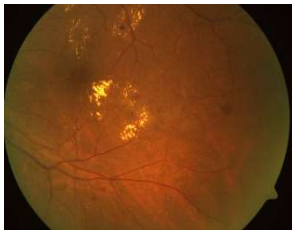


Figure 3.2: Non-Proliferative DR

Figure 3.3: Proliferative DR

3.2 Stochastic Gradient Descent Optimizer(SGD)

SGD is a repeated learning rate optimization algorithm that's been designed specifically for training deep neural networks. It is comparatively a faster optimization technique than the others. SGD carries out attribute upgrade at each step of the training data. These regular upgrade results in fluctuating loss function (FUNC) and hence has an excessive variance. Therefore, it is able to explore other local minima.

$$\theta = \theta - \eta * \Delta(\theta; x(i); y(i)) \tag{3.1}$$

3.3 Batch Normalization

One of the major reasons we use Batch Normalization is to increase the learning rate. A higher learning rate helps to train the model fast. It helps in adjusting weights and also reduces the sensitivity of initial weights assigned to the model.The following are the step carried out by this layer:

• Calculates the variance and the mean of the input layer

• Using previous statistics of the Batch it normalizes the layer

• Output layer is obtained by performing scaling and shifting

3.4 Softmax

Softmax it's a function, not a loss. All the parameters are given a certain weightage. All this weithages add upto one. It is applied to the output scores. As elements represent a class, they can be interpreted as class probabilities. The Softmax function can be computed as:

$$f(s)_i = e^{s_i} / \sum_{c}^{j} e^{s_j} \tag{3.2}$$

CHAPTER 4

SYSTEM DESIGN

4.1 Proposed Work

To develop an automated system for detecting Diabetic Retinopathy Detection, we require a training and testing data set .A model is then developed using this training .The database consists of fundus images of the eye. These images are obtained using the fundus camera. Once obtained, the data is labeled by a professional ophthalmologist.

The labeled data constitutes the training dataset that is required to train the neural network. The training database then undergoes data preprocessing techniques like removal of padding, Gaussian blur, and Feature extraction. The data preprocessing techniques help to eliminate noise and realize the important features of the eye in the fundus image. Due to the small size of the dataset, data Augmentation was performed using Keras ImageDataGenerator. It helps in providing new data by applying various transformations on the training set.

The image dataset is then fed to the Convolutional Neural Network and Resnet for training the model. The model essentially extracts important features and apply commensurate weights. These extracted features hold the key to detecting Diabetic Retinopathy more accurately. The model is then generated which can be used for automated detection of Diabetic Retinopathy. Integrated to a web app using FLASK.

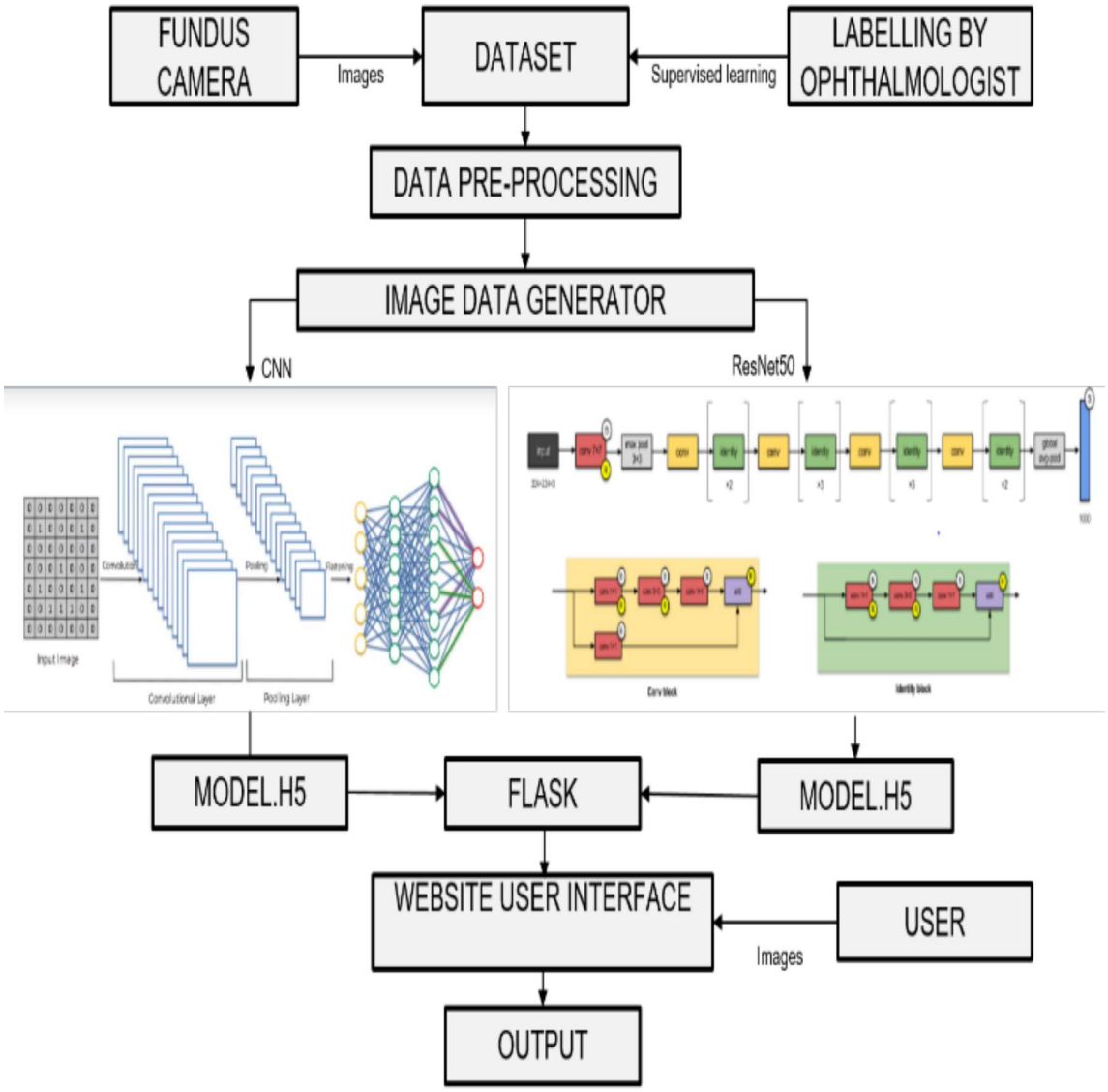


Figure 4.1: Architecture

4.2 Data Pre-Processing

Padding(Removal):

The input data consists of fundus images in various dimensions. The image is encapsulated with noise in the form of padding of blackened pixels. The removal of the padding layer reduces the noise present in the input image while focusing on the essential components of the data. The input data is resized to a predefined size of 256x256 which provides uniformity and stability in applying pre-processing operations on the dataset being fed to the neural network.

Gaussian Blur:

Gaussian Blur is a mathematical function used to extract essential features present in the image. Gaussian Blur helps in blurring the edges and reducing the contrast. It further enhances the essential features of the input data. This helps the algorithm to differentiate between essential and nonessential features present in the image.

Feature Extraction (Circular Crop):

The pre-processed (Gaussian blur filters) image still consists of nonessential components in the form of noise around the corners of the image . Auto cropping is a data pre-processing approach that was applied to the dataset to remove the background noise and realize all the essential feature parameters present in the fundus dataset. Auto Cropping perceives the major portion of the eye required by identifying the circular component of the image where the eye is located. The remaining portion which consisted of nonessential features was blackened out to remove any intrusion to the neural network by the noise. Auto Cropping helps in eradicating the noise as well as extracting all the important features of the eye present in the image.

4.3 Data Augmentation

Data Augmentation plays a vital role in deep learning. Whenever there is insufficient training data we resort to augmenting the data. The fitting of the model often improves with increase in the size of the data set . Data Augmentation is a process that helps proffessionals to enlarge the data set. Using this enlarged data set the model is able to generalize its attributes in a more effective manner. Since the size of the data set is enlarged. it also helps in reducing the chances of Overfitting. Keras provides an inbuilt function that helps in data augmentation called ImageDataGenerator

Keras ImageDataGenerator:

Keras ImageDataGenerator is a data augmentation function that is used to increase the dataset size and helps generate a more normalized input for the network. The training data is fed as a batch of images to the network. The ImageDataGeneraor takes the training data as input which undergoes certain transformations like horizontal_flip, vertical_flip, rotation_range, zoom_range, validation_split, etc. The new set of images along with the previous dataset realizes to be the new dataset for the system improving the normalization of the model.

The Keras ImageDataGenerator class actually works by:

• Taking a set of images from the training data set.

• Here user defined transformations are applied to these set of data.

• The original set of data is replaced by the new set of data

• The training model then trains on the new model.

4.4 Neural Network and Transfer Learning

Convolutional Neural Network(CNN)

CNN is one of the more popular deep learning architectures that are generally utilised in image processing.CNN consists of three distinct layers such as a convolutional layer, pooling layer, and the fully-connected dense layer. All the features are extracted with the use of various convolutional and pooling layer. The high dimensional features which were extracted from the last pooling layer are fed onto the fully connected layer for final optimization. There are various basic building blocks of this architecture namely-

- Convolutional Layer

  This is one of the principal blocks of the CNN Architecture. This layer consists of certain learnable filters. These filters helps in enhancing and mapping various features present in the input image.

- Pooling Layer

  Pooling layer are one of the more fundamental block of the Architecture. There are numerous feature maps that has been extracted by the convolutional layer from the input image. This pooling layer helps in reducing the spacial size, therefore help in reducing the number of attributes as well as computational power.

- ReLu Layer

  Once the image is convolved, the ReLu layer takes this image as an input. This Layer uses a function called max. This MAX function helps in removing all the negative values from the convolved matrix. It uses a functions like tanh or sigmoid.

- Fully Connected Layer

  This layer helps in determining the output. This is the layer that helps in classifying and labelling the image. The decision to classify the image is managed by this layer.

- Loss Layer

  This layer is the last layer that is present in CNN Architecture. This shows the abbreation between predicted and true values. there are various functions that help this layer. we have used the softmax function.

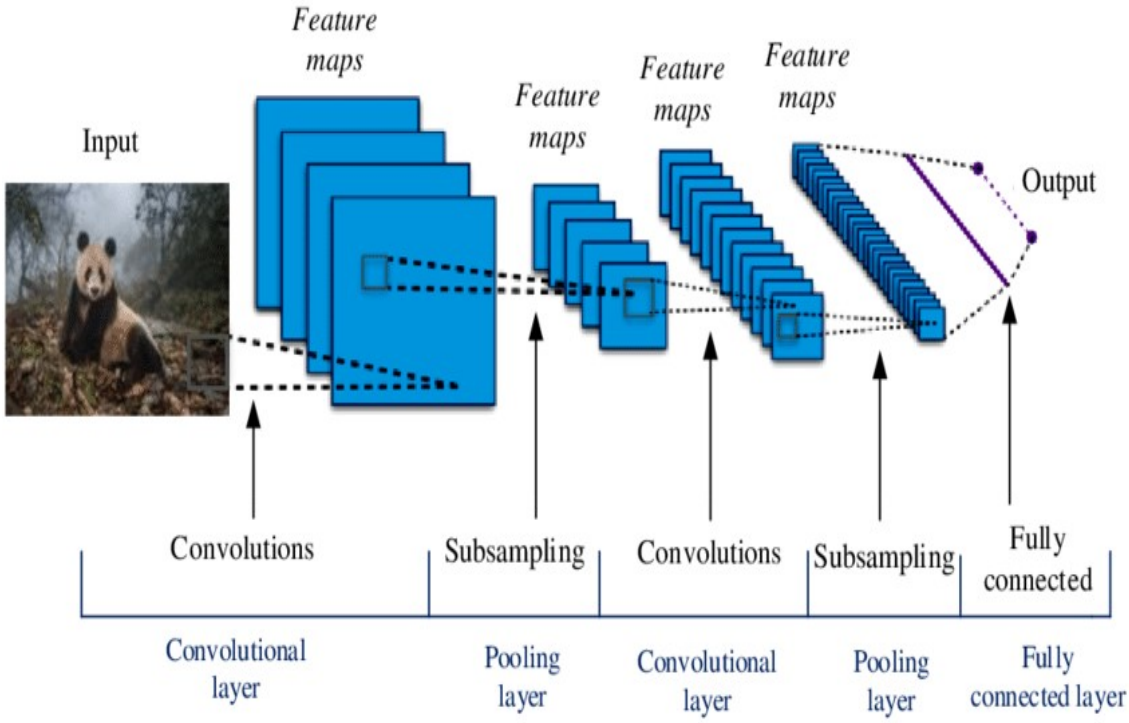


Figure 4.2: CNN ARCHITECTURE

Hyperparameters help us decide the neural structure and determine how the network behaves during training and testing. These parameters are set to optimize the stability and accuracy of the model.

Transfer Learning with ResNet50 Transfer learning is one of the machine learning techniques which helps the model by incorporating model weights file in the neural architecture. It is a widely used deep learning technology which helps in the optimization and improvement of the learning process in a neural network. In a classification model such as ResNet50 reducing computational overhead is very cost-effective transfer learning becomes a major benefit.

4.5 WEB APPLICATION

A web app is useful in providing the patient a platform to test if he or she has DR or not.The web app would also provide certain general information on DR that the user may require. A model will be deployed on this web app with the help of Flask.The patient can upload his/her fundus image of the eye on the web app. This will help the web app predict and give an output to the user.

The image uploaded by the user is first resized to the size that model requires a input layer. The weights saved then help the model in detecting the presence of DR.
The Web App would be Deployed using a flask environment where the backend code would take the input picture from the front end and then predict the class for the skin lesion.
After the prediction for the skin lesion, the result would be returned to the front end where the result would be presented to the user.
Technologies used (**FRONTEND**)-

- HTML
- CSS

Technologies used (**BACKEND**)-

- JAVASCRIPT
- FLASK

CHAPTER 5

CODING

5.1 Importing packages

```
import numpy as np
import pandas as pd
import os
import cv2
import PIL
import gc
import psutil
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.compat.v1 import set_random_seed
from tqdm import tqdm
from math import ceil
import math
import sys
import gc

import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import load_img
from keras.preprocessing.image import array_to_img
from keras.preprocessing.image import img_to_array
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras.models import Model
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.pooling import GlobalAveragePooling2D
from keras.layers import Input
```

```
from keras.layers.core import Dropout
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.callbacks import ModelCheckpoint
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import EarlyStopping

from keras.activations import softmax
from keras.activations import elu
from keras.activations import relu
from keras.optimizers import Adam
from keras.optimizers import RMSprop
from keras.optimizers import SGD
from keras.layers.normalization import BatchNormalization

gc.enable()

print(os.listdir("../input/"))
```

```
['drdataset1', 'drdataset2', 'aptos2019-blindness-detectio
n', 'drmodels', 'drdataset5', 'resnet50weightsfile', 'drda
taset3', 'pateint']


Using TensorFlow backend.
```

5.2 Data Visualisation

```
chat_data = df_train.diagnosis.value_counts()
chat_data.plot(kind='bar');
plt.title('Sample Per Class');
plt.show()
plt.pie(chat_data, autopct='%1.1f%%', shadow=True, labels=["No DR", "
    Non-Proliferative DR (NPDR)", "Proliferative DR (PDR)"])
plt.title('Per class sample Percentage');
plt.show()
```

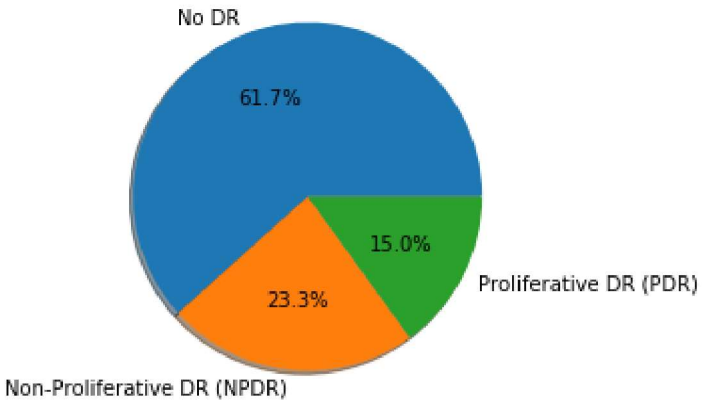


Figure 5.1: Images Per Class

5.3 Conversion to GrayScale

```
figure = plt.figure(figsize=(20, 16))
for target_class in (y_train.unique()):
    for i, (idx, row) in enumerate(
            df_train.loc[df_train.diagnosis == target_class].sample
    (5, random_state=SEED).iterrows()):
        ax = figure.add_subplot(5, 5, target_class * 5 + i + 1)
        imagefile = f"../input/drdataset2/DR/train_image/{row['
    id_code']}.png"
        img = cv2.imread(imagefile)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img = cv2.resize(img, (IMG_DIM, IMG_DIM))
        plt.imshow(img, cmap='gray')
        ax.set_title(CLASSS[target_class])
```

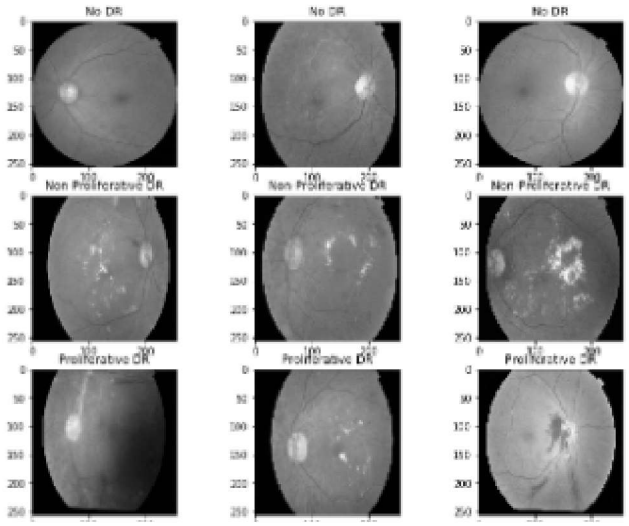


Figure 5.2: Grayscale Image

5.4 Data Prep-processing

Gaussian Blur

```
def draw_img_light(imgs, target_dir, class_label='0'):
    fig, axis = plt.subplots(2, 6, figsize=(15, 6))
    for idnx, (idx, row) in enumerate(imgs.iterrows()):
        imgPath = os.path.join(dir_path, f"{target_dir}/{row['id_code
    ']}.png")
        img = cv2.imread(imgPath)
        row = idnx // 6
        col = idnx % 6
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (IMG_DIM, IMG_DIM))
        img = cv2.addWeighted ( img,4, cv2.GaussianBlur( img , (0,0)
    , IMG_DIM/10) ,-4 ,128) # the trick is to add this line
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        axis[row, col].imshow(img, cmap='gray')
    plt.suptitle(class_label)
    plt.show()
```

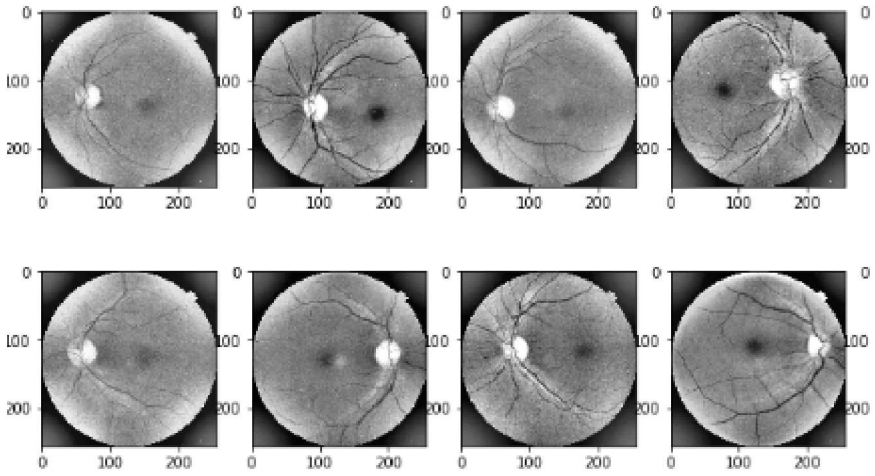


Figure 5.3: Gaussian Blur Images

Removal of Padding

```
def crop_image_from_gray(img,tol=7):
    if img.ndim ==2:
        mask = img>tol
        return img[np.ix_(mask.any(1),mask.any(0))]
    elif img.ndim==3:
        gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        mask = gray_img>tol

        check_shape = img[:,:,0][np.ix_(mask.any(1),mask.any(0))].
    shape[0]
        if (check_shape == 0): # image is too dark so that we crop
    out everything,
            return img # return original image
        else:
            img1=img[:,:,0][np.ix_(mask.any(1),mask.any(0))]
            img2=img[:,:,1][np.ix_(mask.any(1),mask.any(0))]
            img3=img[:,:,2][np.ix_(mask.any(1),mask.any(0))]
    #       print(img1.shape,img2.shape,img3.shape)
            img = np.stack([img1,img2,img3],axis=-1)
    #       print(img.shape)
            return img
```

Feature Extraction(Cropping)

```
def circle_crop(img):
    """
    Create circular crop around eye centre
    """

    #img = cv2.imread(img)
    height, width, depth = img.shape
    largest_side = np.max((height, width))
    img = cv2.resize(img, (largest_side, largest_side))

    x = int(width/2)
    y = int(height/2)
    r = np.amin((x,y))

    circle_img = np.zeros((height, width), np.uint8)
    cv2.circle(circle_img, (x,y), int(r), 1, thickness=-1)
    img = cv2.bitwise_and(img, img, mask=circle_img)

    return img
```

```
def load_ben_color(image, sigmaX=25):
    #image = cv2.imread(path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    #image = cv2.resize(image, (IMG_DIM, IMG_DIM))
    image = crop_image_from_gray(image)
    image = cv2.resize(image, (IMG_DIM, IMG_DIM))
    image = cv2.addWeighted ( image,4, cv2.GaussianBlur( image ,
    (0,0) , sigmaX) ,-4 ,128)
    image = circle_crop(image)
    return image
```

5.5 Data Augmentation

```
datagenerator=ImageDataGenerator(#rescale=1./255,
                                 validation_split=0.15,
                                 horizontal_flip=True,
                                 vertical_flip=True,
                                 #rotation_range=40,
                                 #zoom_range=0.2,
                                 preprocessing_function=
    load_ben_color,
                                 shear_range=0.1,
                                 fill_mode='nearest')
#imgPath = f"../input/aptos2019-blindness-detection/train_images/
    cd54d022e37d.png"
imgPath = f"../input/drdataset2/DR/train_image/02685f13cefd.png"
# Loading image
img = cv2.imread(imgPath)
#img = x_train[0]
img = cv2.resize(img, (IMG_DIM, IMG_DIM))
data = img_to_array(img)
samples =np.expand_dims(data, 0)
i=5
it=datagenerator.flow(samples , batch_size=1)
for i in range(5):
    plt.subplot(230 + 1 + i)
    batch = it.next()
    image = batch[0].astype('uint8')
    plt.imshow(image)
plt.show()
```

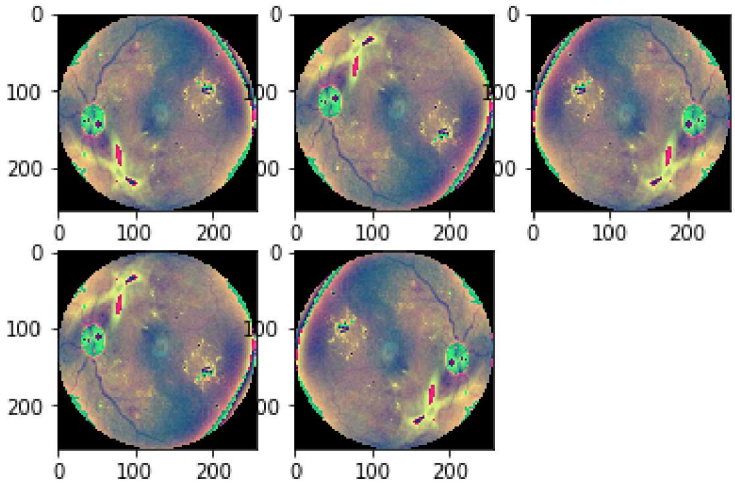


Figure 5.4: Augmented Image

5.6 CNN Model

```
def design_model():
    model = Sequential()
    model.add(Conv2D(filters=16, kernel_size=(2, 2), input_shape=[
    IMG_DIM, IMG_DIM, CHANNEL_SIZE], activation=relu))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(rate=0.2))
    model.add(Conv2D(filters=32, kernel_size=(2, 2), activation=relu)
    )
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(rate=0.2))
    model.add(Conv2D(filters=64, kernel_size=(2, 2), activation=relu)
    )
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(rate=0.2))
    model.add(GlobalAveragePooling2D())
    model.add(Dense(units=256, activation=relu))
    #model.add(Dropout(rate=0.2))
    model.add(Dense(units=512, activation=relu))
    model.add(Dense(3, activation='softmax'))
    return model

gc.collect()
```

```
model = design_model()
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 255, 255, 16)      208
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 127, 127, 16)      0
_________________________________________________________________
dropout_5 (Dropout)          (None, 127, 127, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 126, 126, 32)      2080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 62, 62, 64)        8256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 31, 31, 64)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 31, 31, 64)        0
_________________________________________________________________
global_average_pooling2d_2 ( (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               16640
_________________________________________________________________
dense_5 (Dense)              (None, 512)               131584
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 1539
=================================================================
Total params: 160,307
Trainable params: 160,307
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.5: CNN Model

```
history2 = model.fit_generator(generator=train_generator,
                    validation_data=valid_generator,
                    steps_per_epoch=NUB_TRAIN_STEPS,
                    validation_steps=NUB_VALID_STEPS,
                    verbose=1,
                    #use_multiprocessing=True,
                    #workers=3,
                    callbacks=[early_stop, reduce_lr],
                    shuffle=True,
                    max_queue_size=10,
                    epochs=NUM_EPOCHS)
```

```
Epoch 1/17
94/94 [==============================] - 122s 1s/step - loss: 0.7578 - accuracy: 0.6504 - va
l_loss: 0.6196 - val_accuracy: 0.8281
Epoch 2/17
94/94 [==============================] - 128s 1s/step - loss: 0.3977 - accuracy: 0.8167 - va
l_loss: 0.4248 - val_accuracy: 0.8564
Epoch 3/17
94/94 [==============================] - 127s 1s/step - loss: 0.3635 - accuracy: 0.8194 - va
l_loss: 0.2943 - val_accuracy: 0.8511
Epoch 4/17
94/94 [==============================] - 126s 1s/step - loss: 0.3323 - accuracy: 0.8381 - va
l_loss: 0.3590 - val_accuracy: 0.8830
Epoch 5/17
94/94 [==============================] - 124s 1s/step - loss: 0.3307 - accuracy: 0.8336 - va
l_loss: 0.0961 - val_accuracy: 0.8723
Epoch 6/17
94/94 [==============================] - 123s 1s/step - loss: 0.3441 - accuracy: 0.8292 - va
l_loss: 0.2484 - val_accuracy: 0.8617
Epoch 7/17
94/94 [==============================] - 122s 1s/step - loss: 0.3283 - accuracy: 0.8230 - va
l_loss: 0.3744 - val_accuracy: 0.8670
Epoch 8/17
94/94 [==============================] - 119s 1s/step - loss: 0.3311 - accuracy: 0.8390 - va
l_loss: 0.2644 - val_accuracy: 0.8511

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
Epoch 9/17
94/94 [==============================] - 115s 1s/step - loss: 0.3205 - accuracy: 0.8407 - va
l_loss: 0.3139 - val_accuracy: 0.8777
Epoch 10/17
94/94 [==============================] - 119s 1s/step - loss: 0.2980 - accuracy: 0.8612 - va
l_loss: 0.3381 - val_accuracy: 0.8138
Epoch 11/17
```

```
Epoch 11/17
94/94 [==============================] - 114s 1s/step - loss: 0.3065 - accuracy: 0.8670 - va
l_loss: 0.3293 - val_accuracy: 0.8617

Epoch 00011: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.
Epoch 12/17
94/94 [==============================] - 118s 1s/step - loss: 0.3001 - accuracy: 0.8571 - va
l_loss: 0.3808 - val_accuracy: 0.9096
Epoch 13/17
94/94 [==============================] - 115s 1s/step - loss: 0.3039 - accuracy: 0.8568 - va
l_loss: 0.3540 - val_accuracy: 0.8617
Epoch 14/17
94/94 [==============================] - 112s 1s/step - loss: 0.2833 - accuracy: 0.8621 - va
l_loss: 0.2263 - val_accuracy: 0.8883

Epoch 00014: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-06.
Epoch 15/17
94/94 [==============================] - 121s 1s/step - loss: 0.3003 - accuracy: 0.8470 - va
l_loss: 0.1572 - val_accuracy: 0.8723
```

5.7 Transfer Learning on ResNet50 Model

```
def create_resnet(img_dim, CHANNEL, n_class):
    input_tensor = Input(shape=(img_dim, img_dim, CHANNEL))
    base_model = ResNet50(weights=None, include_top=False,
   input_tensor=input_tensor)
    base_model.load_weights('../input/resnet50weightsfile/
   resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5')
    x = GlobalAveragePooling2D()(base_model.output)
    x = BatchNormalization()(x)
    x = Dropout(0.4)(x)
    x = Dense(2048, activation=elu)(x)
    x = BatchNormalization()(x)
    x = Dropout(0.4)(x)
    x = Dense(1024, activation=elu)(x)
    x = BatchNormalization()(x)
    x = Dropout(0.3)(x)
    x = Dense(512, activation=elu)(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    output_layer = Dense(n_class, activation='softmax', name="
   Output_Layer")(x)
    model_resnet = Model(input_tensor, output_layer)

```

```
    return model_resnet

gc.collect()
model_resnet = create_resnet(IMG_DIM, CHANNEL_SIZE, NUM_CLASSES)
model_resnet.summary()
```

```
------
Layer (type)                 Output Shape         Param #     Connected to
==================================================================================================
======
input_1 (InputLayer)         (None, 256, 256, 3)  0
--------------------------------------------------------------------------------------------------
------
conv1_pad (ZeroPadding2D)    (None, 262, 262, 3)  0           input_1[0][0]
--------------------------------------------------------------------------------------------------
------
conv1 (Conv2D)               (None, 128, 128, 64) 9472        conv1_pad[0][0]
--------------------------------------------------------------------------------------------------
------
bn_conv1 (BatchNormalization) (None, 128, 128, 64) 256        conv1[0][0]
--------------------------------------------------------------------------------------------------
------
activation_1 (Activation)    (None, 128, 128, 64) 0           bn_conv1[0][0]
--------------------------------------------------------------------------------------------------
------
pool1_pad (ZeroPadding2D)    (None, 130, 130, 64) 0           activation_1[0][0]
--------------------------------------------------------------------------------------------------
------
max_pooling2d_1 (MaxPooling2D) (None, 64, 64, 64)   0           pool1_pad[0][0]
--------------------------------------------------------------------------------------------------
------
res2a_branch2a (Conv2D)      (None, 64, 64, 64)   4160        max_pooling2d_1[0][0]
--------------------------------------------------------------------------------------------------
------
bn2a_branch2a (BatchNormalizati (None, 64, 64, 64)   256         res2a_branch2a[0][0]
--------------------------------------------------------------------------------------------------
------
activation_2 (Activation)    (None, 64, 64, 64)   0           bn2a_branch2a[0][0]
--------------------------------------------------------------------------------------------------
------
res2a_branch2b (Conv2D)      (None, 64, 64, 64)   36928       activation_2[0][0]
--------------------------------------------------------------------------------------------------
```

```
batch_normalization_2 (BatchNor (None, 2048)         8192        dense_1[0][0]
__________________________________________________________________________________________________
______
dropout_2 (Dropout)             (None, 2048)         0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
______
dense_2 (Dense)                 (None, 1024)         2098176     dropout_2[0][0]
__________________________________________________________________________________________________
______
batch_normalization_3 (BatchNor (None, 1024)         4096        dense_2[0][0]
__________________________________________________________________________________________________
______
dropout_3 (Dropout)             (None, 1024)         0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
______
dense_3 (Dense)                 (None, 512)          524800      dropout_3[0][0]
__________________________________________________________________________________________________
______
batch_normalization_4 (BatchNor (None, 512)          2048        dense_3[0][0]
__________________________________________________________________________________________________
______
dropout_4 (Dropout)             (None, 512)          0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
______
Output_Layer (Dense)            (None, 3)            1539        dropout_4[0][0]
==================================================================================================
======
Total params: 30,431,107
Trainable params: 30,366,723
Non-trainable params: 64,384
__________________________________________________________________________________________________
```

```
Epoch 1/17
94/94 [==============================] - 136s 1s/step - loss: 1.0722 - accuracy: 0.7651 - va
l_loss: 6.3968 - val_accuracy: 0.2604
Epoch 2/17
94/94 [==============================] - 130s 1s/step - loss: 1.0036 - accuracy: 0.8052 - va
l_loss: 20.0001 - val_accuracy: 0.2766
Epoch 3/17
94/94 [==============================] - 126s 1s/step - loss: 1.0705 - accuracy: 0.7980 - va
l_loss: 6.4289 - val_accuracy: 0.2500
Epoch 4/17
94/94 [==============================] - 130s 1s/step - loss: 1.0997 - accuracy: 0.7847 - va
l_loss: 14.1315 - val_accuracy: 0.6915

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0019999999552965165.
Epoch 5/17
94/94 [==============================] - 130s 1s/step - loss: 1.1069 - accuracy: 0.8185 - va
l_loss: 2.5592 - val_accuracy: 0.2979
Epoch 6/17
94/94 [==============================] - 131s 1s/step - loss: 0.8391 - accuracy: 0.8016 - va
l_loss: 1.4045 - val_accuracy: 0.5106
Epoch 7/17
94/94 [==============================] - 128s 1s/step - loss: 0.7646 - accuracy: 0.8123 - va
l_loss: 0.0877 - val_accuracy: 0.9202
Epoch 8/17
94/94 [==============================] - 127s 1s/step - loss: 0.5585 - accuracy: 0.8381 - va
l_loss: 0.0966 - val_accuracy: 0.9415
Epoch 9/17
94/94 [==============================] - 125s 1s/step - loss: 0.5663 - accuracy: 0.8496 - va
l_loss: 0.2925 - val_accuracy: 0.9202
Epoch 10/17
94/94 [==============================] - 125s 1s/step - loss: 0.5811 - accuracy: 0.8461 - va
l_loss: 0.0304 - val_accuracy: 0.8511
Epoch 11/17
94/94 [==============================] - 124s 1s/step - loss: 0.4311 - accuracy: 0.8585 - va
l_loss: 0.1741 - val_accuracy: 0.9362
```

```
Epoch 12/17
94/94 [==============================] - 123s 1s/step - loss: 0.4954 - accuracy: 0.8514 - va
l_loss: 0.2653 - val_accuracy: 0.8989
Epoch 13/17
94/94 [==============================] - 128s 1s/step - loss: 0.3990 - accuracy: 0.8657 - va
l_loss: 0.1374 - val_accuracy: 0.9309

Epoch 00013: ReduceLROnPlateau reducing learning rate to 0.0003999999724328518.
Epoch 14/17
94/94 [==============================] - 124s 1s/step - loss: 0.3867 - accuracy: 0.8746 - va
l_loss: 0.1934 - val_accuracy: 0.9309
Epoch 15/17
94/94 [==============================] - 124s 1s/step - loss: 0.4529 - accuracy: 0.8621 - va
l_loss: 0.4574 - val_accuracy: 0.9309
Epoch 16/17
94/94 [==============================] - 124s 1s/step - loss: 0.3766 - accuracy: 0.8817 - va
l_loss: 0.0824 - val_accuracy: 0.9309

Epoch 00016: ReduceLROnPlateau reducing learning rate to 7.999999215826393e-05.
Epoch 17/17
94/94 [==============================] - 121s 1s/step - loss: 0.4128 - accuracy: 0.8657 - va
l_loss: 0.2122 - val_accuracy: 0.9096
```

5.8 Website

```
import time
import json
import os
import numpy as np
import pandas as pd
from math import ceil
import cv2
import tensorflow as tf
from flask import Flask, flash, request, redirect, url_for,
    render_template, Session
from werkzeug.utils import secure_filename
import numpy
import matplotlib.pyplot as plt
from io import BytesIO
import base64
import keras
from keras.backend import clear_session
from keras.models import load_model
from keras.optimizers import SGD,RMSprop
from keras.losses import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from flask import jsonify
from tensorflow.compat.v1 import set_random_seed
SEED = 7
np.random.seed(SEED)
set_random_seed(SEED)

graph = tf.get_default_graph()
session = tf.Session(graph = tf.Graph())
with session.graph.as_default():
    keras.backend.set_session(session)
    classifier = load_model('resnet_2048.h5')
    classifier.load_weights('resnet_2048_weights.h5', by_name=True)
    classifier._make_predict_function()
#Pre-processing
def crop_image_from_gray(img,tol=7):
```

```
    if img.ndim ==2:
        mask = img>tol
        return img[np.ix_(mask.any(1),mask.any(0))]
    elif img.ndim ==3:
        gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        mask = gray_img>tol

        check_shape = img[:,:,0][np.ix_(mask.any(1),mask.any(0))].
    shape[0]
        if (check_shape == 0): # image is too dark so that we crop
    out everything,
            return img # return original image
        else:
            img1=img[:,:,0][np.ix_(mask.any(1),mask.any(0))]
            img2=img[:,:,1][np.ix_(mask.any(1),mask.any(0))]
            img3=img[:,:,2][np.ix_(mask.any(1),mask.any(0))]
            img = np.stack([img1,img2,img3],axis=-1)
        return img

def circle_crop(img):

    height, width, depth = img.shape
    largest_side = np.max((height, width))
    img = cv2.resize(img, (largest_side, largest_side))

    x = int(width/2)
    y = int(height/2)
    r = np.amin((x,y))
    circle_img = np.zeros((height, width), np.uint8)
    cv2.circle(circle_img, (x,y), int(r), 1, thickness=-1)
    img = cv2.bitwise_and(img, img, mask=circle_img)
    return img

def load_ben_color(image, sigmaX=10):
    #image = cv2.imread(path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    #image = cv2.resize(image, (IMG_DIM, IMG_DIM))
    image = crop_image_from_gray(image)
    image = cv2.resize(image, (IMG_DIM, IMG_DIM))
```

```
    image = cv2.addWeighted ( image,4, cv2.GaussianBlur( image , (0,0) , sigmaX) ,-4 ,128)
    image = circle_crop(image)
    return image

IMG_DIM = 256

#Load Image
def D(img_path):
    df_class = pd.read_csv("uploader/2.csv")
    #df_class.id_code = df_class.id_code.apply(lambda x: x + ".png")
    df_class['id_code'] = df_class['id_code'].astype('str')
    #classifier.summary()
    #Single Prediction
    test_img = cv2.imread(img_path)
    test_img = cv2.resize(test_img,(IMG_DIM,IMG_DIM))
    test_img = np.expand_dims(test_img, axis=0)
    dummy_datagen=ImageDataGenerator(rescale=1./255, preprocessing_function=load_ben_color)
    #dummy_generator = dummy_datagen.flow(test_img, y=None, batch_size=1, seed=7)
    dummy_generator = dummy_datagen.flow_from_dataframe(dataframe= df_class,
                                                        #directory="../input/aptos2019-blindness-detection/train_images/",
                                                        directory="uploader/",
                                                        x_col="id_code",
                                                        #y_col="diagnosis",
                                                        #y_col=["diagnosis_0","diagnosis_1","diagnosis_2"],
                                                        batch_size=1,
                                                        class_mode=None,
                                                        target_size=(IMG_DIM, IMG_DIM),
                                                        #shuffle=False,
                                                    )
    with session.graph.as_default():
        keras.backend.set_session(session)
```

```
        tta_steps = 5
        preds_tta = []
        for i in range(tta_steps):
            dummy_generator.reset()
            preds = classifier.predict_generator(generator=
    dummy_generator, steps=ceil(df_class.shape[0]))
            preds_tta.append(preds)
        final_pred = np.mean(preds_tta, axis=0)
        predicted_class_indices = np.argmax(final_pred, axis=1)

        #Label Dictionary
        label_maps = {0: 'No DR', 1: 'Non-Proliferative DR', 2: '
    Proliferative DR'}
        label = label_maps[int(predicted_class_indices)]
        return (label)


i=1
j=2
UPLOAD_FOLDER = 'uploader/'

app = Flask(__name__)
app.secret_key = 'some_secret'
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1


app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

@app.after_request
def add_header(response):
    # response.cache_control.no_store = True
    response.headers['Cache-Control'] = 'no-store, no-cache, must-
    revalidate, post-check=0, pre-check=0, max-age=0'
    response.headers['Pragma'] = 'no-cache'
    response.headers['Expires'] = '-1'
    return response

@app.route("/")
def home():
```

```
  return render_template("index.html")

@app.route("/Info")
def Info():
    return render_template('Info.html')

@app.route("/DR", methods=['GET', 'POST'])
def DR():
    return render_template('base.html')

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'GET':
        return render_template('base.html')
    else:
        file = request.files['image']
        file.save(os.path.join('uploader', "2.png"))
        res = D('uploader/2.png')
        return str(res)
    return None




if __name__ == "__main__":
  app.run(host='127.0.0.1', port=5001, debug=True)
```

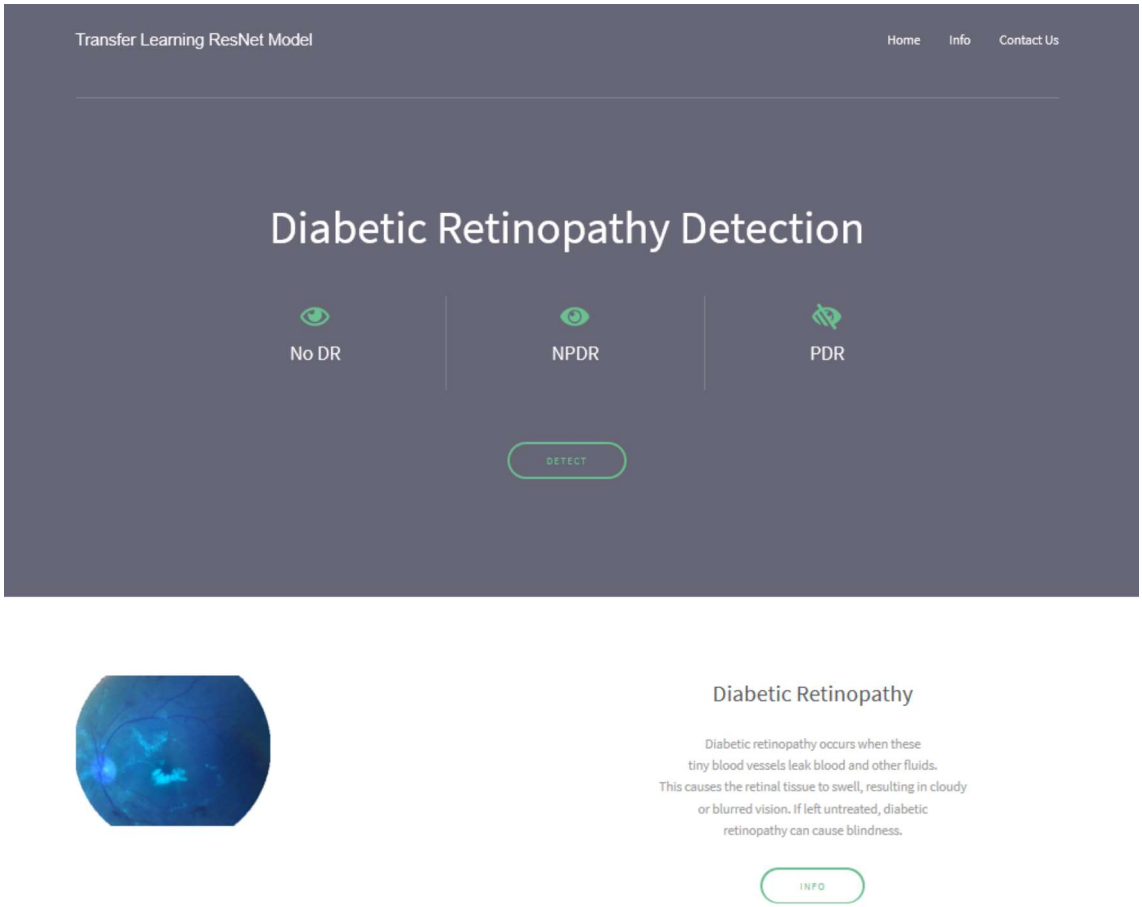


Figure 5.6: homepage

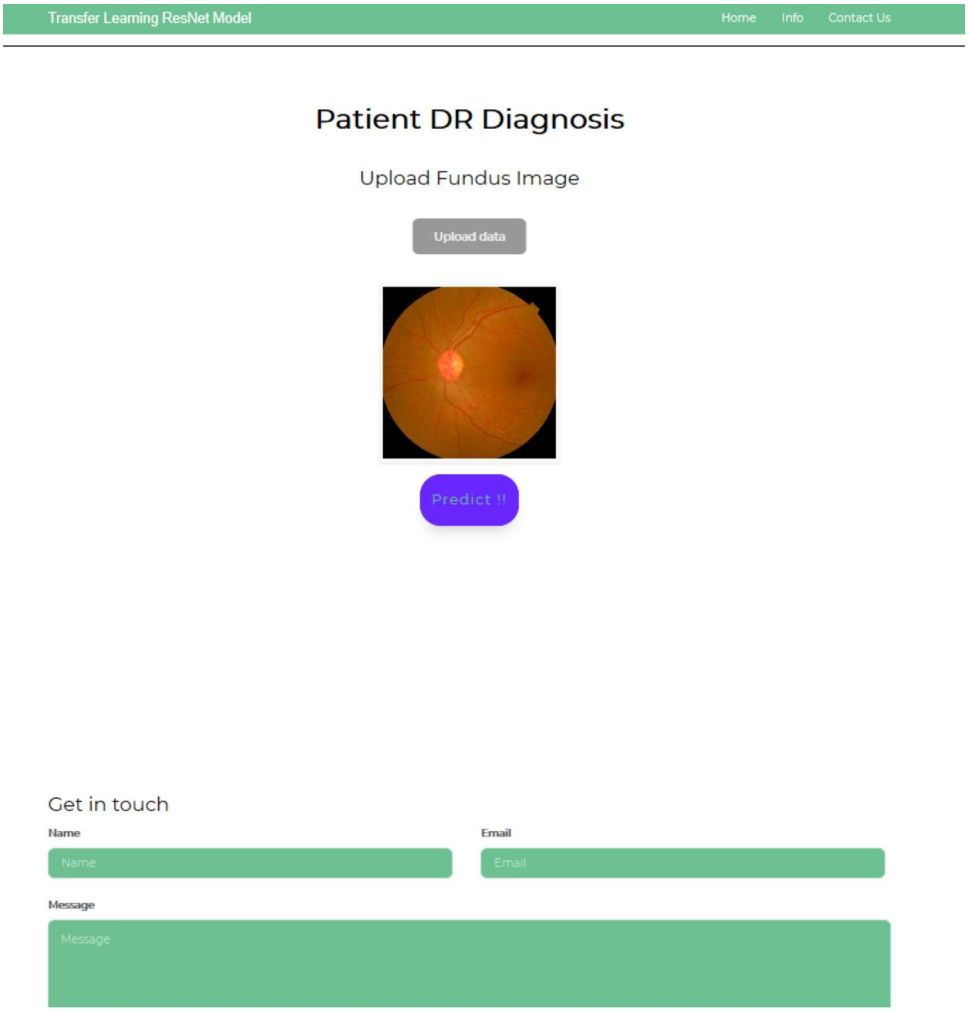


Figure 5.7: Upload page

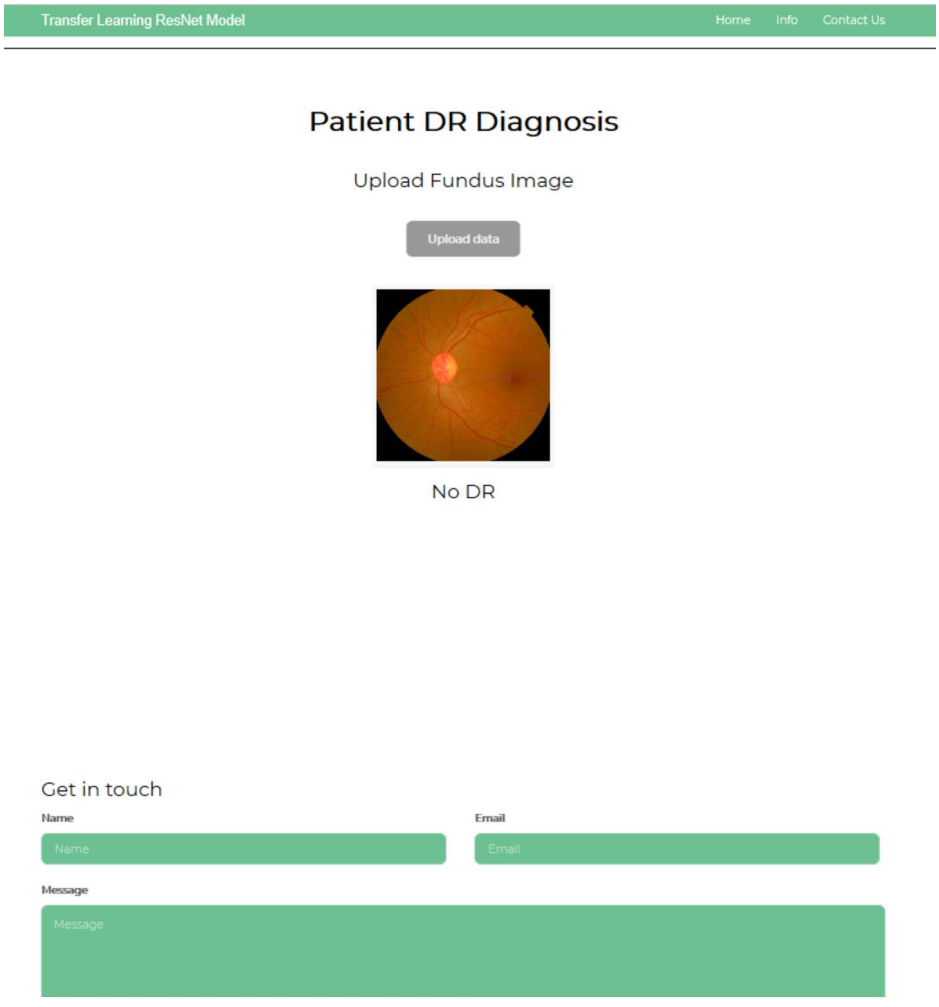


Figure 5.8: Prediction

CHAPTER 6

RESULTS DISCUSSION

In this paper, the proposed methodology contains a series of simple yet effective preprocessing techniques like removal of padding, auto-cropping and Gaussian blur which has shown great effect.

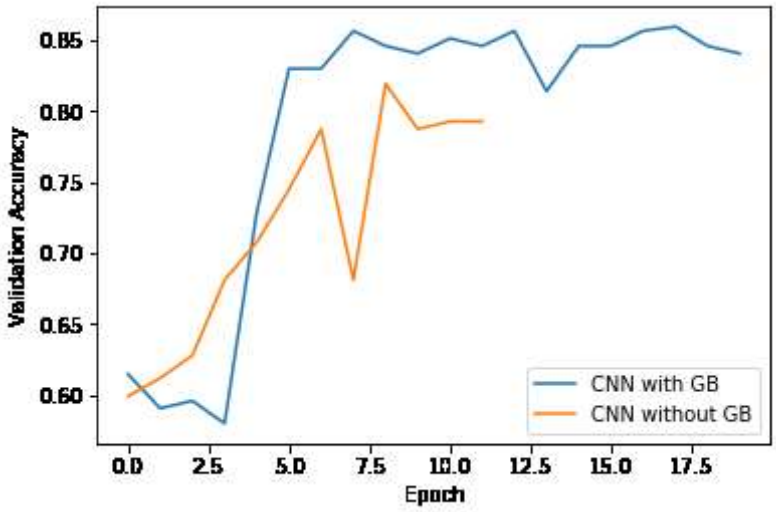


Figure 6.1: Comparison of CNN with v/s without Gaussian Blur

As seen with the accuracy curve of the CNN model, from the comparison of CNN layers with and without Gaussian Blur where the former came out to be triumphed by a significant improvement in the overall stability and precision of the model. Auto cropping and Gaussian Blur removes the irrelevant portions captured in a fundus image and keeps the vital part of the eye to be used for the detection and classification of the Diabetic retinopathy. As seen with the accuracy curve of the CNN model, Gaussian Blur helps to improve the accuracy of Diabetic Retinopathy detection. CNN was only able to achieve about 81% but with Gaussian blur added as a preprocessing technique, the model was able to achieve an accuracy of about 86%. This shows how valuable Gaussian blur is as a preprocessing technique for Fundus Images. The loss score obtained when using CNN architecture without Gaussian Blur is approximately 0.18 whereas when Gaussian Blur was applied we were able to achieve a loss score of 0.14.

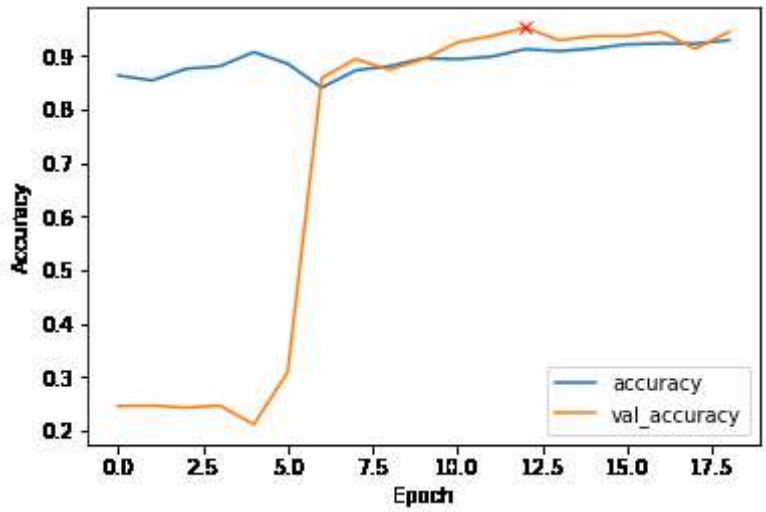


Figure 6.2: RESNET50 ACCURACY CURVE

ResNet50 was the most successful architecture. ResNet was able to achieve this with the help of transfer learning which greatly reduces computation time and other preprocessing techniques which resulted in an optimal score of 96.16%. The loss score attained using transfer Learning with ResNet50 is 0.0002.

Table 6.1: COMPARISON OF MODELS USED

NAME	PERFORMANCE		
	VALIDATION ACC	TEST ACC	LOSS FUNC
CNN without Gaussian Blur	81.91%	69.19%	0.189
CNN with Gaussian Blur	86.16%	78.67%	0.1425
RESNET50 withGaussian Blur	93.9%	92.99%	0.0002

After applying further optimization on classification, the specificity of PDR is about 99.1%, MPDR is approximately 92.3% and that of NPDR is 93.4%. This shows that it is capable of detecting Diabetic Retinopathy in real-time.

CHAPTER 7

CONCLUSION

The results achieved in this study are consistent with the other works that have been done with deep convolutional neural networks. The results obtained by our proposed model are in accordance with the current res and display that better results are obtained by models that are deeper. The results verify the proposal that features learned by pre-trained models help to learn features for a completely different domain dataset, in our case the Diabetic Retinopathy images dataset. The use of transfer learning models for feature extraction is a high performance-yielding technique for medical image analysis. Additionally, fine-tuning and data augmentation are important parameters for a better model.

CHAPTER 8

REFERENCES

[1] Gupta.A and Chhikarab.R, "Diabetic Retinopathy: Present and Past", Procedia Computer Science Volume 132, 2018, Pages 1432-1440.

[2] Wu.Y and Hu.Z," Recognition of Diabetic Retinopathy Based on Transfer Learning", 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analytics.

[3] Yu.S, Xiao.D, and Kanagasingam.Y, " Exudate Detection for Diabetic Retinopathy With Convolutional Neural Networks ", 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)

[4] Shirbahadurkar.S.D, Mane.V.M and Jadhav.D.V, " A Modern Screening Approach for Detection of Diabetic Retinopathy", 2017 2nd International Conference on Man and Machine Interfacing (MAMI)

[5] Chandran.A, Prof. Nisha K K, and Dr. Vineetha, "Computer-Aided Approach for Proliferative Diabetic Retinopathy detection in color retinal images", 2016 International Conference on Next Generation Intelligent Systems (ICNGIS)

[6] Arora.M and Pandey.M, " Deep Neural Network for Diabetic Retinopathy Detection ", 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con), India, 14th -16th Feb 2019.

[7] Tjandrasa.H, Putra.R.E and Wijaya.A.Y, Arieshanti.I, " Classification of Non-Proliferative Diabetic Retinopathy Based on Hard Exudates Using Soft Margin SVM", 2013 IEEE International Conference on Control System, Computing and Engineering

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

[9] Yadav.J, Sharma.M, and Saxena.V, " Diabetic Retinopathy Detection using feed-forward Neural Network", 2017 Tenth International Conference on Contemporary Computing (IC3)

[10] Dr. R. GeethaRamani, Shanthamalar.J and Lakshmi, "Automatic Diabetic Retinopathy Detection through Ensemble Classification Techniques", 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)

[11] https://images.app.goo.gl/q7M716STojgnqfy58,https://www.researchgate.net/profile/Ayseguel_I example-of-a-simple-CNN-architecture.png

APPENDIX A

OTHER ESSENTIAL CODES

- **Feeding Data**

```
    df_train = pd.read_csv(os.path.join(dir_path, "train.csv"))
df_test = pd.read_csv(os.path.join(dir_path, "test.csv"))
NUM_CLASSES = df_train['diagnosis'].nunique()

```

- **Splitting Data set**

```
        x_train, x_test, y_train, y_test = train_test_split(df_train.id_code, df_train.diagnosis, test_size=0.15,
                                                            random_state=SEED, stratify=df_train.diagnosis)

```

- **Finding maximum and minimum height and width**

```
        def check_max_min_img_height_width(df, img_dir):
    max_Height , max_Width =0 ,0
    min_Height , min_Width =sys.maxsize ,sys.maxsize
    for idx, row in df.iterrows():
        imgPath=os.path.join(dir_path,f"{img_dir}/{row['id_code']}.png")
        img=cv2.imread(imgPath)
        H,W=img.shape[:2]
        max_Height=max(H,max_Height)
        max_Width =max(W,max_Width)
        min_Height=min(H,min_Height)
        min_Width =min(W,min_Width)
    return max_Height, max_Width, min_Height, min_Width

```

- **Accuracy curve plot**

```
plt.figure(figsize=(8, 8))
plt.title("Learning curve")
plt.plot(history1.history["loss"], label="loss")
plt.plot(history1.history["val_loss"], label="val_loss")
plt.plot(np.argmin(history1.history["val_loss"]), np.min(history1.history["val_loss"]), marker="x", color="r",
         label="best model")
plt.xlabel("Epochs")
plt.ylabel("log_loss")
plt.legend();

```

CHAPTER 9

PLAGIARISM REPORT

Format - I

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

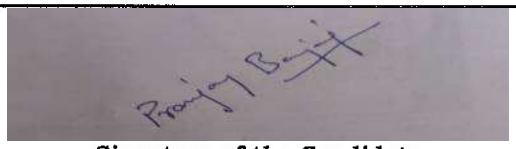
Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES
(To be attached in the dissertation/ project report)

1	Name of the Candidate (IN BLOCK LETTERS)	Pranjay Bajaj
2	Address of the Candidate	G-103,Keerthi Harmony Apt,T.C Palaya Road, Ramamurthy Nagar, Bangalore-560016 Mobile Number : 8147119291
3	Registration Number	RA1611003011005
4	Date of Birth	25/11/1998
5	Department	Computer Science And Engineering(CSE)
6	Faculty	
7	Title of the Dissertation/Project	Diabetic Retinopathy Detection Using Fundus Photography
8	Whether the above project/dissertation is done by	~~Individual~~ or group : (Strike whichever is not applicable) a) If the project/ dissertation is done in group, then how many students together completed the project : 2 b) Mention the Name & Register number of other candidates :Hritik Rao RA1611003010365
9	Name and address of the Supervisor / Guide	Mrs. Kanmani Sivagar kanmanis@srmist.edu.in Mail ID : Mobile Number : 9952097187
10	Name and address of the Co-Supervisor / Co- Guide (if any)	Mail ID : Mobile Number :

11	Software Used	Turnitin		
12	Date of Verification	3 April 2020		
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the Chapter**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (Excluding self citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	INTRODUCTION	0%	0%	0%
2	LITERATURE SURVEY	1%	1%	1%
3	SYSTEM ANALYSIS	0%	0%	0%
4	SYSTEM DESIGN	0%	0%	0%
5	CODING	5%	5%	5%
6	RESULTS DISCUSSION	0%	0%	0%
7	CONCLUSION	0%	0%	0%
8				
9				
10				
Appendices		0%	0%	0%

I / We declare that the above information have been verified and found true to the best of my / our knowledge.



Signature of the Candidate

Name & Signature of the Staff (Who uses the plagiarism check software)

Name & Signature of the Supervisor/Guide

Name & Signature of the Co-Supervisor/Co-Guide

Name & Signature of the HOD

ORIGINALITY REPORT

13%	5%	5%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to B.S.Abdur Rahman Crescent Institute of Science & Technology Student Paper	4%
2	Submitted to Indian School of Business Student Paper	3%
3	Submitted to University of Oxford Student Paper	1%
4	Submitted to SRM University Student Paper	1%
5	Submitted to UT, Dallas Student Paper	1%
6	Submitted to Monash University Student Paper	1%
7	www.business-school.ed.ac.uk Internet Source	1%
8	Submitted to University of Hull Student Paper	1%
9	Submitted to Taylor's Education Group	

CHAPTER 10

LIST OF PUBLICATIONS

International Journal of Innovative Technology and Exploring Engineering (IJI-TEE)

International Journal of Innovative Technology and Exploring Engineering

ISSN: 2278-3075 (Online) | Exploring Innovation | A Key for Dedicated Services

Published by: Blue Eyes Intelligence Engineering and Sciences Publication

G18-19-20, Block-B, Tirupati Abhinav Homes, Damkheda, Bhopal (Madhya Pradesh)-462037, India

Website: www.ijitee.org Email: submit2@ijitee.org

CERTIFICATE

This certifies that the research paper entitled **'Diabetic Retinopathy Detection using Fundus Photography'** authored by **'Hritik Rao, Pranjay Bajaj, Kanmani Sivagar'** was reviewed by experts in this research area and accepted by the board of 'Blue Eyes Intelligence Engineering and Sciences Publication' which has published in **'International Journal of Innovative Technology and Exploring Engineering (IJITEE)'**, ISSN: 2278–3075 (Online), Volume-9 Issue-6, April 2020, Page No. 1194-1198.

The Value of Citation (VoC): of IJITEE is 6.03 for year 2019. Your published paper and Souvenir are available at: https://www.ijitee.org/download/volume-9-issue-6/

Jitendra Kumar Sen
(Manager)

Blue Eyes Intelligence Engineering and Sciences Publication
BHOPAL

Dr. Shiv Kumar
(CEO)

International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075, Volume-9 Issue-6, April 2020

Diabetic Retinopathy Detection using Fundus Photography

Hritik Rao, Pranjay Bajaj, Kanmani Sivagar

Abstract*— Patients suffering from prolonged diabetic conditions are prone to Diabetic Retinopathy (DR) which leads to vision impairment if left untreated. Diabetic Retinopathy has been on the rise across the globe due to an increase in the number of diabetic patients. Diabetic Retinopathy detection in early stages has become vital to prevent permanent vision impairment and avoid arduous medical treatment in the later stages. Diabetic Retinopathy (DR) causes damage to the retina and gradual loss of sight and in severe cases permanent vision impairment eventually leading to blindness. An early analysis of Diabetic Retinopathy helps in controlling the progress of the disease and increases the chances of recovery. An automated classification of Diabetic Retinopathy using images is a difficult job due to the microscopic variability of the appearance of different classes and the lack of a standard data infrastructure by medical professionals. One of the major deterrents in automated Diabetic Retinopathy (DR) detection is the identification of the essential features in the fundus image. Techniques like Gaussian Blur and auto-cropping has been used for feature extraction and noise removal. Through this paper, we aim to classify various fundus images of the eye into various classes of diabetic Retinopathy and automate the screening process.

Keywords*— Diabetic Retinopathy, Retina, fluorescein angiography, Convolutional Neural Network, radial,** **Optical coherence tomography,** ***preprocessing.

I. INTRODUCTION

Over the years there has been a drastic increase in the number of people affected by diabetes. Diabetes is complemented by various other diabetic conditions which generally affect people who have had extended suffering from diabetes. Diabetic Retinopathy is a condition caused by perpetual suffering from diabetes. It is the major leading cause of blindness in this present day and age. It has affected almost 93 million people in the world. It causes gradual loss of sight which leads to blindness in the long run.

All kinds of diabetic patients are affected by diabetic retinopathy in due course. The patients suffering from type 1 diabetes generally tend to feel the effects of DR at a later stage whereas in the case of type 2 diabetic patients it is the contrary. Diabetic Retinopathy causes the blood vessels present in the eye to leak blood and other fluids around the retina. As the severity of Diabetic Retinopathy increases, it causes the blood vessels to get blocked hence the retina stops receiving proper nourishment. The retina starts to become unhealthy which eventually leads to loss of sight.

Revised Manuscript Received on March 30, 2020.

Hritik Rao, Student, Department of Computer Science & Engineering, SRM Institute of Science and Technology, Kattankulathur India.

Pranjay Bajaj, Student, Department of Computer Science & Engineering, SRM Institute of Science and Technology, Kattankulathur India.

Kanmani Sivagar, Assistant Professor, Department of Computer Science & Engineering, SRM Institute of Science and Technology, Chennai, India.

Early diagnosis is of the essence in the treatment of Diabetic Retinopathy. If we can analyze the Diabetic Retinopathy early, then there is a great possibility of containing the disease and hence avoiding further escalation. Conventional methods to detect diabetic retinopathy require regular screening of the eye. In the current scenario, there is a lot of expert manpower required for the present screening facilities that are needed for this purpose. Therefore, there is a need to place an automated screening technique in its place. By doing this it will drastically reduce the workload and give quicker outcomes and hence plays a helping hand to the ophthalmologists.

With advancements in machine learning, various algorithms are being used to generate an automated system to carry out the screening process. However, in recent times, various techniques such as CNN, transfer learning have brought in drastic changes in the field of medical science. Researchers have started utilizing deep learning as a method to perform image segmentation and classification of Diabetic Retinopathy.

Deep learning is a method that emulates the functioning of the human brain which gathers all the data and creates a knowledge base. Using this knowledge it extracts various information like patterns and takes decisions on it. The deep learning model is first trained on a certain set of labeled and structured databases and then tested onto the other set of databases. Hence, deep learning helps in deploying a computer model that helps in classifying data into different categories. These data could be anything from sounds to texts to images.

Deep learning model has proven to have high computational powers as well as has been able to extract a multitude of features from the given labeled dataset. Hence various deep learning methods like convolutional neural networks are being used in various medical institutes, banking, and corporate world. Hence deep learning algorithms have had a greater impact in today's classification and detection model of image processing, text processing or sound processing systems.

II. LITERATURE SURVEY

The current methodology of research employs various machine learning algorithms for the diagnosis of diseases along with the help of available medical records for the classification and detection of various diseases. An intensive survey was carried out to gain knowledge on the methods applied by researchers and to expand our knowledge of various preprocessing methods. This survey also helped us know about various data augmentation procedures that could help in getting more generalized datasets.

The approach used by Ankita Gupta and Rita Chhikarab [1] divides their detection of DR into two different approaches. The first approach focuses on Blood Vessel Segmentation and the second on the Identification of lesions. This paper helps us in understanding and gaining knowledge about various results like sensitivity, AUC, the accuracy that was derived when different machine learning algorithms were used such as Improved Match Filtering, Ensemble Classifier, etc.

The algorithm applied by Yuchen Wu and Ze Hu [2] uses the Migration Learning Approach which is one of the new machine learning approaches. There are four different approaches to implement this method. They have used Feature-based Transfer Learning. The pre-training model they used is based on ImageNet. Since there was a requirement of data augmentation which involved the usage of imageDataGenerator which is a class present in Keras. Hence Keras was primarily used. They used the pre-training model for feature extraction that was required to develop the final model to detect DR.

The results obtained by using CNN has been projected by the paper published by Shuang Yu, Di Xiao and Yogesan Kanagasingam [3] It uses pixel-wise exudate image patches. The accuracy that was obtained was off 91.92%. The Framework for exudate detection with deep learning includes three main procedures before the image is sent to the DR model that was established using CNN. The three major steps are Removal of Optic Disc Detection, Removal of Retinal Vessels, and Ultimate Opening. The images are entered in the form of 64*64 patches.

There have been various modern screening approaches that have been used to diagnose Diabetic Retinopathy. S.D. Shirbahadurkar, V. M. Mane and D. V. Jadhav[4] have tried to use decision trees to diagnose DR. This algorithm can classify the fundus image into various DR. Hence this algorithm is holoentropy enabled. The above method uses activities like grey scaling (converts image into a pixel driven image that a computer understands), optic disc segmentation and blood vessel segmentation to detect DR. The following steps include feature extraction and hence a feature extraction vector to which appropriate weights are assigned to different features which help in detecting DR.

There also have been various approaches for detecting proliferative DR. This research was published by Anaswara Chandran, Prof. Nisha K K and Dr. Vineetha S [5]. The ability to handle higher dimensional feature sets was essential for the classification process. Hence, Random-Forest Classifier was used here. Different patches of the dataset are used to train various decision trees. The patches are extracted from the image sets. The patches undergo two distinct extraction processes namely Texture Feature Extraction and Vessel Feature Extraction. The decision trees after giving various decisions its output is fed forward to random forest classifier which uses a rule-based approach to classify the images to reach a certain result.

The research methodology employed by Mamta Arora and Mrinal Pandey of Manav Rachna University [6] shows the various stages involved in using deep neural networks for the diagnosis of diabetic retinopathy detection. They followed a two-step process which includes step 1: data preprocessing and augmentation and step2: convolution layer. The convolutional layer was further divided into a 5 step process that involved the convolutional layer which is the basic building block of Convolutional neural extracting weights, Pooling Layer, connecting layer and logistic classifier.

Placing emphasis on non-proliferative DR, SVM algorithm was used for the diagnoses. This is shown by Handayani Tjandrasa et al. [7] and Enrique V. Carrera et al. [8]. The main features of this paper were exudate segmentation, feature extraction and the classification of NPDR severity level. The extracted features were finally trained and tested using soft margin SVM as a classification model.

III. PROPOSED WORK

The overall framework for Diabetic Retinopathy detection has been illustrated in fig 1. The database consists of fundus images of the eye. These images are obtained using the fundus camera. Once obtained, the data is labeled by a professional ophthalmologist. The labeled data constitutes the training dataset that is required to train the neural network. The training database then undergoes data preprocessing techniques like removal of padding, Gaussian blur, and Feature extraction. The data preprocessing techniques help to eliminate noise and realize the important features of the eye in the fundus image. Due to the small size of the dataset, data Augmentation was performed using Keras ImageDataGenerator. It helps in providing new data by applying various transformations on the training set. The image dataset is then fed to the Convolutional Neural Network and Resnet for training the model. The model essentially extracts important features and apply commensurate weights. These extracted features hold the key to detecting Diabetic Retinopathy more accurately. The model is then generated which can be used for automated detection of Diabetic Retinopathy. Integrated to a web app using FLASK.

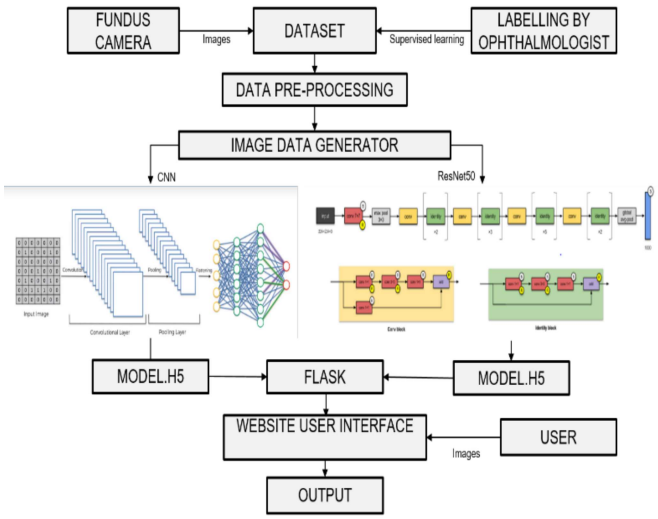


Figure 1. Architecture

A. *Data Introduction*

The dataset consists of fundus images of the eye which were obtained through various medical professionals.

For the development of dataset, macula-centered fundus images were obtained from EYEPACS which is a U.S. based private organization and three other hospitals in India. The dataset consisted of fundus images from various patients with varying levels of illumination and other physical parameters leading to inconsistencies with the data even in the same class. The dataset consists of three categorical classes of Diabetic Retinopathy. The classes being No DR, Non-proliferative DR, proliferative DR.

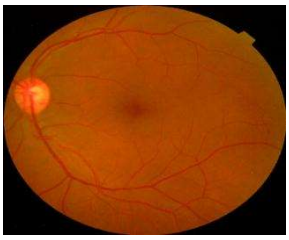


Figure 2 NO DR

Figure 3 Non-Proliferative DR

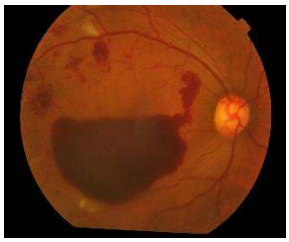


Figure 4 Proliferative DR

B. *Data Preprocessing*

Padding (Removal):

The input data consists of fundus images in various dimensions. The image is encapsulated with noise in the form of padding of blackened pixels. The removal of the padding layer reduces the noise present in the input image while focusing on the essential components of the data. The input data is resized to a predefined size of 256x256 which provides uniformity and stability in applying pre-processing operations on the dataset being fed to the neural network.

Gaussian Blur:

Gaussian Blur is a mathematical function used to extract essential features present in the image. Gaussian Blur helps in blurring the edges and reducing the contrast. It further enhances the essential features of the input data. This helps the algorithm to differentiate between essential and nonessential features present in the image.

Feature Extraction (Circular Crop):

The pre-processed (Gaussian blur filters) image still consists of nonessential components in the form of noise around the corners of the image due to the spherical shape of the eyeball. Auto cropping is a data pre-processing approach that was applied to the dataset to remove the background noise and realize all the essential feature parameters present in the fundus dataset. Auto Cropping perceives the major portion of the eye required by identifying the circular component of the image where the eye is located. The remaining portion which consisted of nonessential features was blackened out to remove any intrusion to the neural network by the noise. Auto Cropping helps in eradicating the noise as well as extracting all the important features of the eye present in the image.

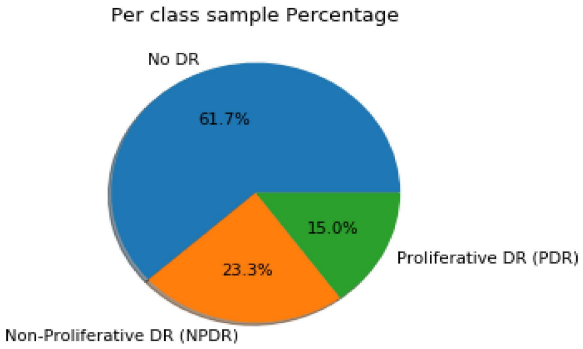


Figure 5 Dataset in each class

C. *Data Augmentation*

Keras ImageDataGenerator is a data augmentation function that is used to increase the dataset size and helps generate a more normalized input for the network. The training data is fed as a batch of images to the network. The ImageDataGeneraor takes the training data as input which undergoes certain transformations like horizontal_flip, vertical_flip, rotation_range, zoom_range, validation_split, etc. The new set of images along with the previous dataset realizes to be the new dataset for the system improving the normalization of the model.

D. *Neural Network and Transfer Learning*

Convolutional Neural Network:

CNN consists of three distinct layers such as a convolutional layer, pooling layer, and the fully-connected dense layer. All the features are extracted with the use of various convolutional and pooling layer. The high dimensional features which were extracted from the last pooling layer are fed onto the fully connected layer for final optimization.

Hyperparameters help us decide the neural structure and determine how the network behaves during training and testing. These parameters are set to optimize the stability and accuracy of the model.

Transfer Learning with ResNet50

Transfer learning is one of the machine learning techniques which helps the model by incorporating model weights file in the neural architecture. It is a popular deep learning technology which helps in the optimization and improvement of the learning process in a neural network. In a classification model such as ResNet50 reducing computational overhead is very cost-effective transfer learning becomes a major benefit.

IV. EXPERIMENTAL RESULTS

In this paper, the proposed methodology contains a series of simple yet effective preprocessing techniques like removal of padding, auto-cropping and Gaussian blur which has shown a great effect.

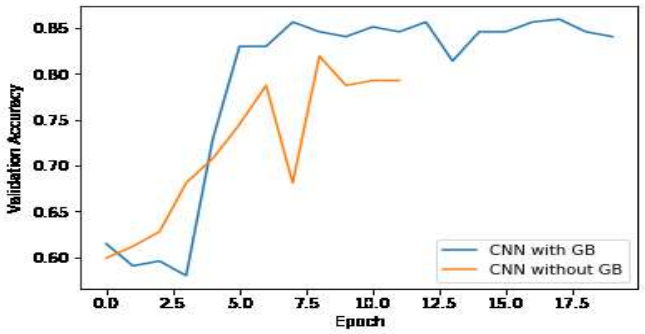


Figure 6 Comparison

As seen with the accuracy curve of the CNN model, from the comparison of CNN layers with and without Gaussian Blur where the former came out to be triumphed by a significant improvement in the overall stability and precision of the model. Auto cropping and Gaussian Blur removes the irrelevant portions captured in a fundus image and keeps the vital part of the eye to be used for the detection and classification of the Diabetic retinopathy.

As seen with the accuracy curve of the CNN model, Gaussian Blur helps to improve the accuracy of Diabetic Retinopathy detection. CNN was only able to achieve about 81% but with Gaussian blur added as a preprocessing technique, the model was able to achieve an accuracy of about 86%. This shows how valuable Gaussian blur is as a preprocessing technique for Fundus Images. The loss score obtained when using CNN architecture without Gaussian Blur is approximately 0.18 whereas when Gaussian Blur was applied we were able to achieve a loss score of 0.14.

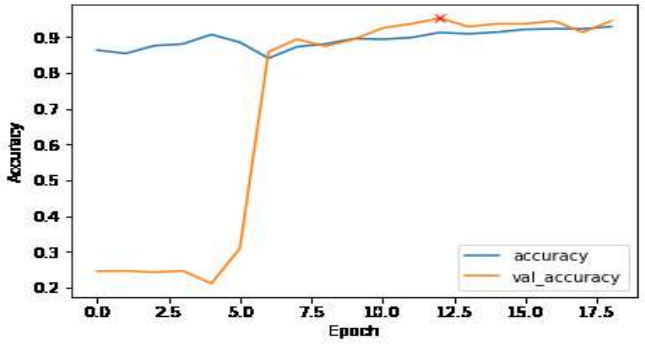


Figure 7 ResNet50 Accuracy

ResNet50 was the most successful architecture. ResNet was able to achieve this with the help of transfer learning which greatly reduces computation time and other preprocessing techniques which resulted in an optimal score of 96.16%. The loss score attained using transfer Learning with ResNet50 is 0.0002.

Name	CNN Model Without Gaussian Blur	CNN model with Gaussian Blur	ResNet50 Model with Gaussian Blur
Validation Accuracy	81.91%	86.16%	94.15%
Testing Accuracy and Loss function	69.19% & 0.1893	78.67% & 0.1425	93.99% & 0.0002

After applying further optimization on classification, the specificity of PDR is about 99.1%, MPDR is approximately 92.3% and that of NPDR is 93.4%. This shows that it is capable of detecting Diabetic Retinopathy in real-time.

V. CONCLUSION

The results achieved in this study are consistent with the other works that have been done with deep convolutional neural networks. The results obtained by our proposed model are in accordance with the current res and display that better results are obtained by models that are deeper. The results verify the proposal that features learned by pre-trained models help to learn features for a completely different domain dataset, in our case the Diabetic Retinopathy images dataset. The use of transfer learning models for feature extraction is a high performance-yielding technique for medical image analysis. Additionally, fine-tuning and data augmentation are important parameters for a better model.

REFERENCES

1. Ankita Gupta and Rita Chhikarab, "Diabetic Retinopathy: Present and Past", Procedia Computer Science Volume 132, 2018, Pages 1432-1440.
2. Yuchen Wu and Ze Hu," Recognition of Diabetic Retinopathy Based on Transfer Learning", 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analytics.
3. Shuang Yu, Di Xiao, and Yogesan Kanagasingam, " Exudate Detection for Diabetic Retinopathy with Convolutional Neural Network",2017 39th Annual International Conference of the IEEE.
4. S.D. Shirbahadurkar, V. M. Mane and D. V. Jadhav, " A Modern Screening Approach for Detection of Diabetic Retinopathy", 2017 2nd International Conference on Man and Machine Interfacing (MAMI).
5. Anaswara Chandran, Prof. Nisha K K, and Dr. Vineetha, "Computer-Aided Approach for Proliferative Diabetic Retinopathy detection in color retinal images", 2016 International Conference on Next Generation Intelligent Systems.
6. Mamta Arora and Mrinal Pandey, " Deep Neural Network for Diabetic Retinopathy Detection ", 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con), India, 14th -16th Feb 2019.
7. Handayani Tjandrasa, Ricky Eka Putra and Arya Yudhi Wijaya, Isye Arieshanti" Classification of Non-Proliferative Diabetic Retinopathy Based on Hard Exudates Using Soft Margin SVM", 2013IEEE International Conference on Control System, Computing and Engineering.
8. Enrique V. Carrera, Andrés González, and Ricardo Carrera, "Automated detection of diabetic retinopathy using SVM", 2017 IEEE XXIV International Conference on Electronics, Electrical .
9. Jayant Yadav, Manish Sharma, and Vikas Saxena, " Diabetic Retinopathy Detection using feedforward Neural Network", 2017 Tenth International Conference on Contemporary Computing (IC3),
10. Vaishali Suryawanshi and Shilpa Setpal, " Gaussian transformed glcm features for Classifying Diabetic Retinopathy", 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS).
11. Dr. R. GeethaRamani, Jeslin Shanthamalar and Lakshmi, "Automatic Diabetic Retinopathy Detection through Ensemble Classification Techniques", 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).

AUTHOR'S PROFILE

Hritik Rao, an aspiring data scientist, currently pursuing his B. Tech in Computer Science and Engineering from SRM Institute of Science and Technology, Kattankulathur. His areas of interest for research include Machine Learning, Medical Artificial Intelligence, and Computer Vision.

Pranjay Bajaj, an aspiring data analyst, pursuing his B. Tech in Computer Science and Engineering from SRM Institute of Science and Technology, Kattankulathur. His research interests include software engineering Machine Learning and, Medical Artificial Intelligence.

Kanmani Sivagar, completed her M. Tech in Computer Science and Engineering in 2010. Currently,she is working as Assistant Professor in Department of Computer Science & Engineering at SRM Institute of Science and Technology, Chennai. Her research interests are in artificial Intelligence, with an emphasis on improving and developing automated systems . Published many research papers in Indian and International journals.