Research Internship Report

Project titled

Diabetic Retinopathy Detection Using Deep Learning

by

Darshan N Shenoy *20BLC1076*

Bachelor of Technology

In

Electronics and Computer Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Submitted To

Dr. Sathiya Narayanan

July 2023

About Dataset:

The dataset used for training the model is sourced from Kaggle website. The link to the dataset is given below-

https://www.kaggle.com/datasets/tanlikesmath/diabetic-retinopathy-resized

The dataset contains-

- Resized Train
  This folder contains around 36000 images of eyes of patients having diabetic retinopathy of different levels (0-4).
- Train Labels
  This contains a csv file with labels of the above images. Labels depict the level of intensity of the disease.

Code:

```
import tensorflow as tf
tf.__version__
```

```
# Import The Libraries

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.resnet50 import ResNet50,
preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img
from tensorflow.keras.models import Sequential

import numpy as np
import pandas as pd
from glob import glob
import matplotlib.pyplot as plt
```

```
data =
pd.read_csv("C:/Users/Admin/Documents/20BLC1076/trainLabels.csv")
data.head()
```

```
data['image_name'] = [i+".jpeg" for i in data['image'].values]
data.head()
```

```
from sklearn.model_selection import train_test_split
```

```
train, val = train_test_split(data, test_size=0.2)
```

```
train.shape, val.shape
```

```
from keras.preprocessing.image import ImageDataGenerator
```

```
import cv2
def load_ben_color(image):
    IMG_SIZE = 224
    sigmaX=10
    image = cv2.resize(image, (IMG_SIZE, IMG_SIZE))
    image=cv2.addWeighted ( image,4, cv2.GaussianBlur( image , (0,0) ,
sigmaX) ,-4 ,128)
    return image

data_gen = ImageDataGenerator(rescale=1/255.,
                              zoom_range=0.15,
                              fill_mode='constant',
                              cval=0.,
                              horizontal_flip=True,
                              vertical_flip=True,
                              preprocessing_function=load_ben_color)


# batch size
bs = 128

train_gen = data_gen.flow_from_dataframe(train,
                                         "C:/Users/Admin/Documents/20BL
C1076/resized_train/resized_train",
                                         x_col="image_name",
y_col="level", class_mode="raw",
```

```
                                                batch_size=bs,
                                                target_size=(224, 224))
val_gen = data_gen.flow_from_dataframe(val,
                                       "C:/Users/Admin/Documents/20BLC1
076/resized_train/resized_train",
                                       x_col="image_name",
y_col="level", class_mode="raw",
                                       batch_size=bs,
                                       target_size=(224, 224))


resnet = tf.keras.applications.resnet.ResNet152(
    include_top=False,
    weights='imagenet',
    input_shape=(224,224,3)
    )

resnet.summary()

import keras.layers as L
from keras.models import Model
from keras.callbacks import EarlyStopping

for layer in resnet.layers:
    layer.trainable = False


x = resnet.output
y = resnet.output

#Pooling Layer
x = L.GlobalMaxPooling2D()(x)
y = L.GlobalAveragePooling2D()(y)

#Flattening Layer
x = L.Flatten()(x)
y = L.Flatten()(y)

#Batch Normalization
x = L.BatchNormalization()(x)
y = L.BatchNormalization()(y)

#Concatenation
output = tf.keras.layers.concatenate(
    [x, y], axis=1       #concatenate along row axis.
)
```

```
output = L.Dropout(0.25)(output)

output = L.Dense(1024, activation="relu")(output)
output = L.Dense(512, activation="relu")(output)
output = L.Dropout(0.50)(output)

output = L.Dense(256, activation="relu")(output)
output = L.Dense(128, activation="relu")(output)
output = L.Dropout(0.50)(output)

output = L.Dense(64, activation="relu")(output)
predictions = L.Dense(5, activation='softmax')(output)


print(predictions)



tf.data.experimental.enable_debug_mode()


model = Model(inputs=resnet.input, outputs=predictions)


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])


callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=10, verbose=1)




history = model.fit(
    train_gen,
    validation_data = val_gen,
    epochs = 50,
    steps_per_epoch = 220,
    validation_steps = 55,
    callbacks = [callback],
    verbose = 1
)
```

```
# Retrieve the training and validation loss values
train_loss = history.history['loss']
val_loss = history.history['val_loss']

# Retrieve the training and validation accuracy values
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

# Plot the loss curves
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss Curves')
plt.legend()

# Plot the accuracy curves
plt.subplot(1, 2, 2)
plt.plot(train_acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy Curves')
plt.legend()

# Display the plot
plt.tight_layout()
plt.show()
```

```
from tensorflow.keras.models import Model

# Create a new model that extracts features
feature_extractor = Model(inputs=model.input, outputs=model.layers[-
5].output)
```

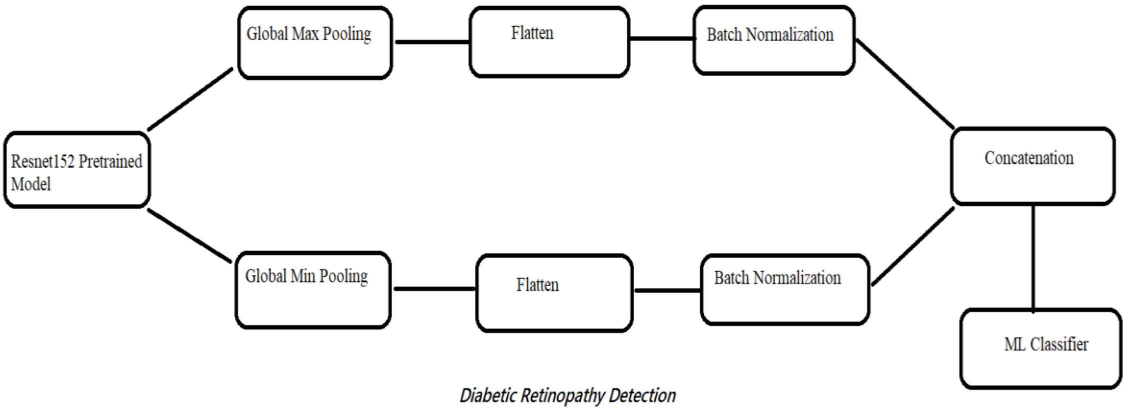
```
train_features = feature_extractor.predict(train_gen)
```

```
val_features = feature_extractor.predict(val_gen)

train_features = train_features.reshape(train_features.shape[0], -1)
val_features = val_features.reshape(val_features.shape[0], -1)



from sklearn.svm import SVC

svm = SVC()
svm.fit(train_features, train['level'])



# Evaluate the SVM classifier

svm_predictions = svm.predict(val_features)
accuracy = np.sum(svm_predictions == val['level']) / len(val)
print("SVM Accuracy:", accuracy)




from sklearn.neighbors import KNeighborsClassifier


knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(train_features, train['level'])


# Evaluate the KNN classifier
svm_predictions = svm.predict(val_features)
accuracy = np.sum(svm_predictions == val['level']) / len(val)
print("KNN Accuracy:", accuracy)
```

Summary:

- Import libraries to use in this project work which include TensorFlow, NumPy, Pandas and others.
- Load the “TrainLabels.csv” dataset into the model. Subsequently add a new column to the existing dataset named “image_name.” It consists of names of images with an extension of “.jpeg.”

- Split the dataset into Training and Validation set in the ratio 80:20.
- Then images are processed based on requirements of the model. For example, image is resized to 224*224 pixel which is a requirement of Resnet Model.
- Image Data is generated using "ImageDataGenerator" function of the TensorFlow library. Batch Size is set to 128.



Diabetic Retinopathy Detection Model Architecture

- This is the architecture that I have implemented for detection of diabetic retinopathy. The architecture consists of Pre-trained Resnet model, pooling layers, Flattening layers, Batch Normalization layers, Concatenation and finally a ML Classifier layer.
- It is using Adam as the optimizer. It has an Early Stopping with a patience of 10. Then we run this model for 50 epochs.
- Finally, we run it through classifiers like SVM or KNN to get the final output.

Result:

In [3]:
```
data = pd.read_csv("C:/Users/Admin/Documents/20BLC1076/trainLabels.csv")
data.head()
```

Out[3]:

	image	level
0	10_left	0
1	10_right	0
2	13_left	0
3	13_right	0
4	15_left	1

- This is brief overview of trainLabels.csv dataset. It consists of image and level columns.

In [4]:
```
data['image_name'] = [i+".jpeg" for i in data['image'].values]
data.head()
```

Out[4]:

	image	level	image_name
0	10_left	0	10_left.jpeg
1	10_right	0	10_right.jpeg
2	13_left	0	13_left.jpeg
3	13_right	0	13_right.jpeg
4	15_left	1	15_left.jpeg

- A new column named “image_name” was added to trainLabels.csv dataset with image names having an extension of “.jpeg.”

```
In [7]: train.shape, val.shape
Out[7]: ((28100, 3), (7026, 3))
```

- Training and validation dataset is divided in the ratio 80:20. Therefore out of 35,126 datapoints, 28,100 datapoints form the training dataset and 7,206 datapoints form the validation datapoints.

```
# batch size
bs = 128

train_gen = data_gen.flow_from_dataframe(train,
                                         "C:/Users/Admin/Documents/20BLC1076/resized_train/resized_train",
                                         x_col="image_name", y_col="level", class_mode="raw",
                                         batch_size=bs,
                                         target_size=(224, 224))
val_gen = data_gen.flow_from_dataframe(val,
                                       "C:/Users/Admin/Documents/20BLC1076/resized_train/resized_train",
                                       x_col="image_name", y_col="level", class_mode="raw",
                                       batch_size=bs,
                                       target_size=(224, 224))

Found 28100 validated image filenames.
Found 7026 validated image filenames.
```

- The batch size is set to 128. The batch size is the number of samples processed before the model is updated.

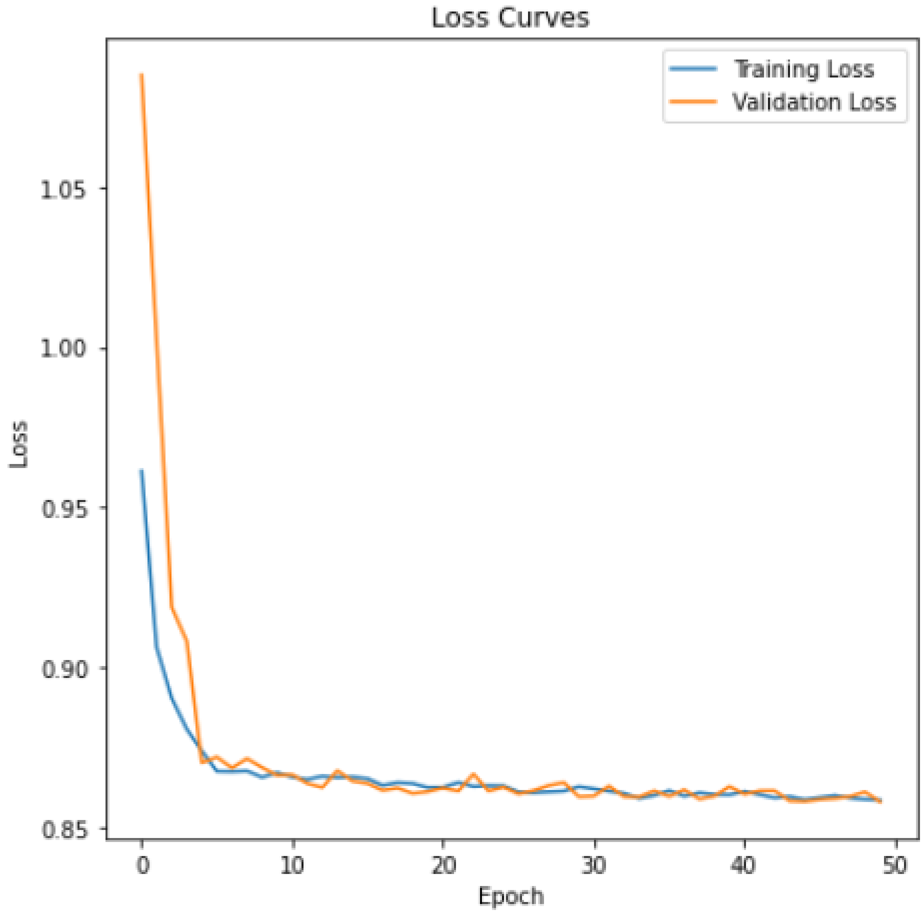
```
In [13]: resnet.summary()

Model: "resnet152"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']
```

- A pre-trained model, Resnet152 is used as the base model for training our model. It contains 152 layers in total and the above is the summary of the pre-trained model.

```
Epoch 1/50
220/220 [==============================] - 2678s 12s/step - loss: 0.9613 - accuracy: 0.7254 - val_loss: 1.0849 - val_accurac
y: 0.7348
Epoch 2/50
220/220 [==============================] - 2684s 12s/step - loss: 0.9062 - accuracy: 0.7347 - val_loss: 1.0021 - val_accurac
y: 0.7348
Epoch 3/50
220/220 [==============================] - 2684s 12s/step - loss: 0.8903 - accuracy: 0.7348 - val_loss: 0.9188 - val_accurac
y: 0.7348
Epoch 4/50
220/220 [==============================] - 2673s 12s/step - loss: 0.8807 - accuracy: 0.7348 - val_loss: 0.9081 - val_accurac
y: 0.7348
Epoch 5/50
220/220 [==============================] - 2671s 12s/step - loss: 0.8738 - accuracy: 0.7347 - val_loss: 0.8701 - val_accurac
y: 0.7348
Epoch 6/50
220/220 [==============================] - 2661s 12s/step - loss: 0.8675 - accuracy: 0.7348 - val_loss: 0.8719 - val_accurac
y: 0.7348
```

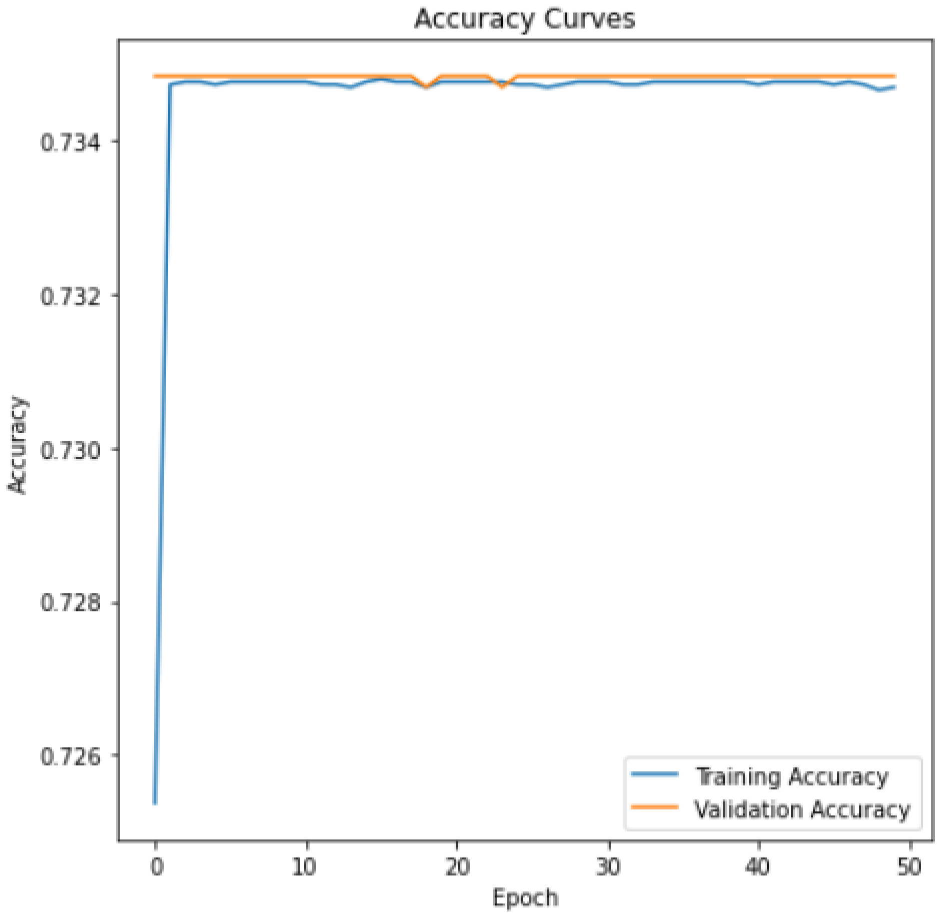
```
Epoch 45/50
220/220 [==============================] - 2655s 12s/step - loss: 0.8586 - accuracy: 0.7348 - val_loss: 0.8581 - val_accurac
y: 0.7348
Epoch 46/50
220/220 [==============================] - 2651s 12s/step - loss: 0.8593 - accuracy: 0.7347 - val_loss: 0.8587 - val_accurac
y: 0.7348
Epoch 47/50
220/220 [==============================] - 2661s 12s/step - loss: 0.8599 - accuracy: 0.7348 - val_loss: 0.8590 - val_accurac
y: 0.7348
Epoch 48/50
220/220 [==============================] - 2651s 12s/step - loss: 0.8591 - accuracy: 0.7347 - val_loss: 0.8596 - val_accurac
y: 0.7348
Epoch 49/50
220/220 [==============================] - 2654s 12s/step - loss: 0.8587 - accuracy: 0.7347 - val_loss: 0.8610 - val_accurac
y: 0.7348
Epoch 50/50
220/220 [==============================] - 2651s 12s/step - loss: 0.8585 - accuracy: 0.7347 - val_loss: 0.8580 - val_accurac
y: 0.7348
```

- The model was run over 50 epochs. An epoch is when all the training data is used at once and is defined as the total number of iterations of all the training data in one cycle for training the deep learning model.
- Early stopping function was also used by setting monitor as validation loss and with a patience of 10.



- A loss function is a mathematical function that quantifies the difference between predicted and actual values in a machine learning model.
  It measures the model's performance and guides the optimization process by providing feedback on how well it fits the data.

- The lower the loss, the better is the model. The loss curve shows a steady decline in the both training and validation loss.



- The Accuracy score is calculated by dividing the number of correct predictions by the total prediction number. The training and validation accuracy of the model has saturated at around 73.48%.