Early Detection of Alzheimer's Disease using CNN Model

Problem Statement:

Alzheimer's disease is a progressive brain disorder that causes memory loss and cognitive decline. Early diagnosis of Alzheimer's is critical for better disease management and treatment. Currently, doctors rely on a combination of neurological exams, cognitive tests, and brain imaging techniques to diagnose Alzheimer's disease. However, these diagnostic methods can be time-consuming, expensive, and may not be accurate in the early stages of the disease.

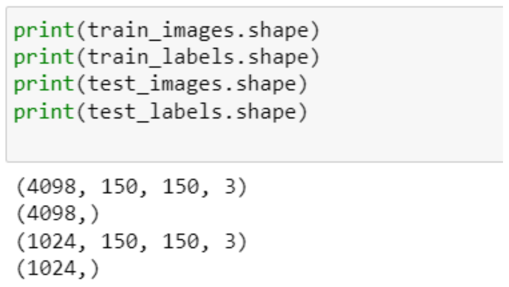
Solution Proposed:

Convolutional neural networks (CNNs) have shown promising results in medical image analysis and can aid in the early detection and accurate diagnosis of Alzheimer's disease. By training a CNN model on a large dataset of MRI brain scans from both healthy individuals and those with Alzheimer's disease, we can develop a tool that can help clinicians accurately diagnose Alzheimer's disease from brain scans. This could lead to earlier diagnosis and better management of the disease, ultimately improving patient outcomes.

Dataset Description:

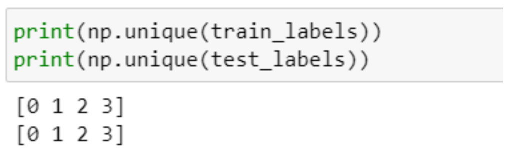
The given data set consists of MRI (Magnetic Resonance Imaging) images. The data is divided into two sets, one for training and the other for testing. The images are classified into four different classes, namely Mild Demented, Moderate Demented, Non Demented, and Very Mild Demented. These classes correspond to different stages of dementia, a neurodegenerative disorder that affects cognitive function and memory. The Mild Demented and Moderate Demented classes indicate the presence of significant cognitive decline, while the Non Demented class represents a normal cognitive function, and the Very Mild Demented class indicates a very early stage of cognitive decline. This information is important for understanding the nature of the data and can help in designing appropriate machine learning models to classify the MRI images into their respective categories.

The below image shows the shape of the train and test dataset.



```
print(train_images.shape)
print(train_labels.shape)
print(test_images.shape)
print(test_labels.shape)
```

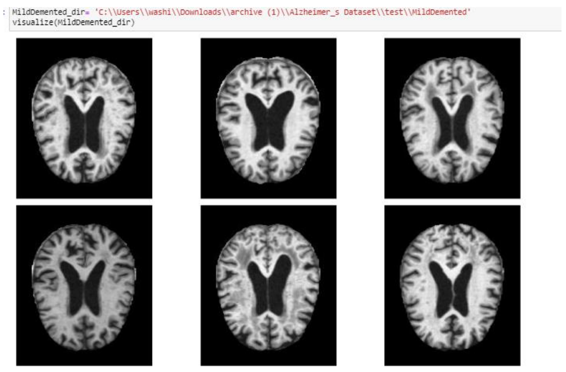
```
(4098, 150, 150, 3)
(4098,)
(1024, 150, 150, 3)
(1024,)
```



```
print(np.unique(train_labels))
print(np.unique(test_labels))
```

```
[0 1 2 3]
[0 1 2 3]
```

Sample MRI scan image dataset:



```
MildDemented_dir= 'C:\\Users\\washi\\Downloads\\archive (1)\\Alzheimer_s Dataset\\test\\MildDemented'
visualize(MildDemented_dir)
```

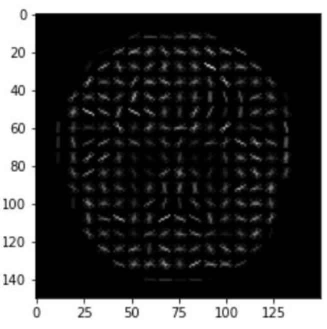
Handling Missing Value:

In image processing using Convolutional Neural Networks (CNNs), missing values can be a common issue that needs to be addressed. For each .jpg file, it reads the image using the Image module from the PIL library and converts it into a NumPy array using the np.array() function. It then checks if the array contains any NaN (not a number) values using the np.isnan() function.

If it does contain NaN values, it appends the filename to a list called missing_values.

Feature Extraction:

In the context of Alzheimer's disease diagnosis using MRI images, feature extraction plays a crucial role in accurately identifying and classifying brain images. One popular method of feature extraction is the Histogram of Oriented Gradients (HOG), which is a technique that captures local gradients of an image and uses them as features. We have used HOG to extract important features in the images. The below image shows important features from a sample training MRI scan.

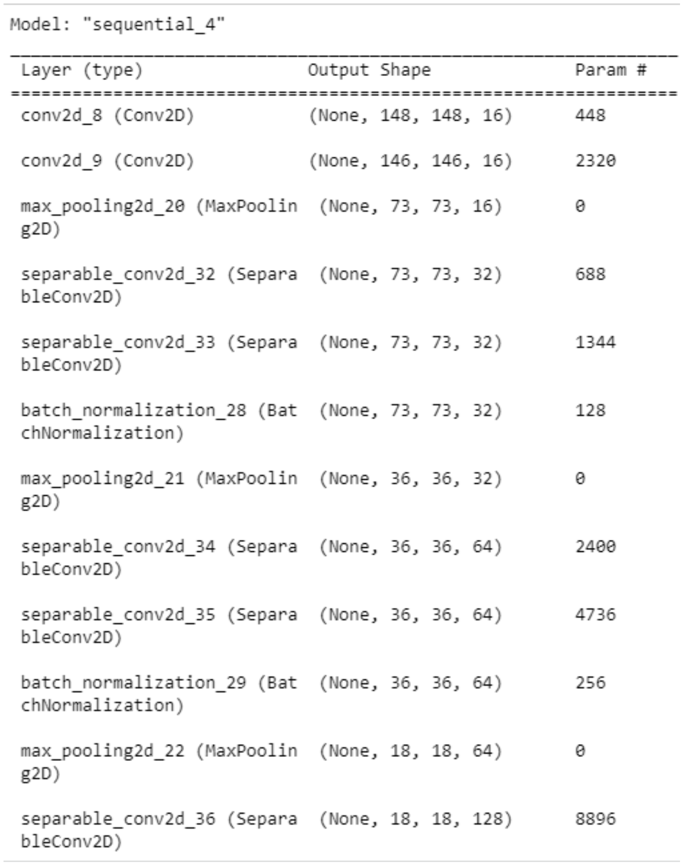


Model Layers:

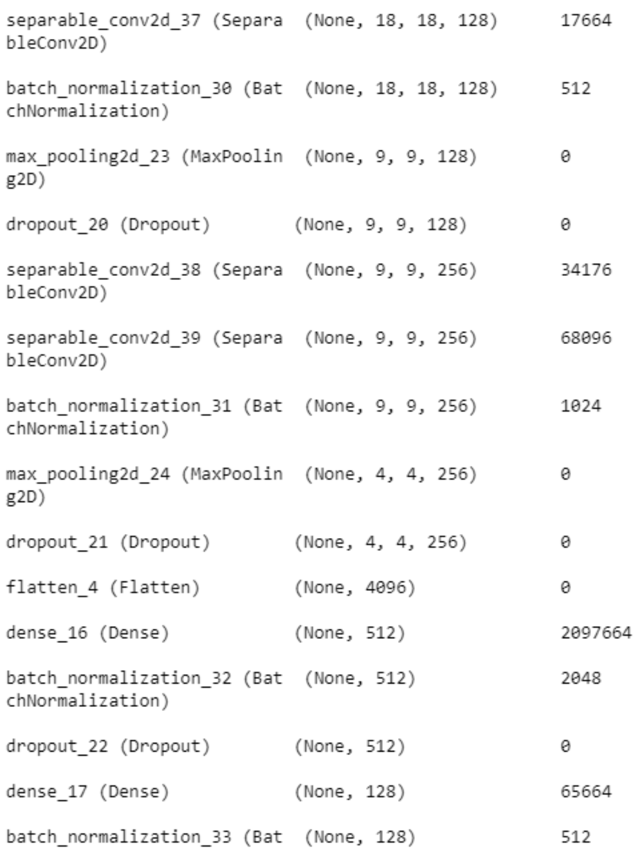
We have defined convolutional neural network (CNN) for image classification. The model starts with a 2D convolutional layer with 16 filters, each of size 3x3, using the ReLU activation function. The input shape for the first layer is 150x150x3, which represents the height, width, and color channels of the image. This is followed by another 2D convolutional layer with the same configuration, then a max pooling layer with a size of 2x2. The model continues with several separable convolutional layers and dropout layers with a rate of 0.2. The final layers include a fully connected layer with 512 nodes and ReLU activation, followed by batch normalization and dropout with a rate of 0.7. Another fully connected layer with 128 nodes and ReLU activation is added, followed by batch normalization and dropout with a rate of 0.5. The model also has two additional fully connected layers with 64 nodes and ReLU activation, followed by batch normalization and dropout with a rate of 0.3, and a final dense layer with 4 nodes and softmax activation, which represents the output classes of the model. The below table shows the description of the layers we used in our model.

Activation Function	Output Size	No. of Parameters	No. of Filters
Relu	(148, 148, 16)	448	16
Relu	(146, 146, 16)	2320	16
MaxPooling2D	(73, 73, 16)	0	0
Relu	(73, 73, 32)	1376	32
Relu	(73, 73, 32)	2080	32
BatchNormalization	(73, 73, 32)	128	0
MaxPooling2D	(36, 36, 32)	0	0
Relu	(36, 36, 64)	4160	64
Relu	(36, 36, 64)	8256	64
BatchNormalization	(36, 36, 64)	256	0
MaxPooling2D	(18, 18, 64)	0	0
Dropout	(18, 18, 64)	0	0
relu	(18, 18, 256)	16640	256
relu	(18, 18, 256)	65792	256
BatchNormalization	(18, 18, 256)	1024	0
MaxPooling2D	(9, 9, 256)	0	0
Dropout	(9, 9, 256)	0	0
Flatten	20736	0	0
relu	512	10619904	0
BatchNormalization	512	2048	0
Dropout	512	0	0
relu	128	65664	0
BatchNormalization	128	512	0
Dropout	128	0	0
relu	64	8256	0
BatchNormalization	64	256	0
Dropout	64	0	0
Dense (softmax)	4	260	0

<u>Output of model Summary:</u>



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 148, 148, 16)      448

 conv2d_9 (Conv2D)           (None, 146, 146, 16)      2320

 max_pooling2d_20 (MaxPoolin  (None, 73, 73, 16)       0
 g2D)

 separable_conv2d_32 (Separa  (None, 73, 73, 32)       688
 bleConv2D)

 separable_conv2d_33 (Separa  (None, 73, 73, 32)       1344
 bleConv2D)

 batch_normalization_28 (Bat  (None, 73, 73, 32)       128
 chNormalization)

 max_pooling2d_21 (MaxPoolin  (None, 36, 36, 32)       0
 g2D)

 separable_conv2d_34 (Separa  (None, 36, 36, 64)       2400
 bleConv2D)

 separable_conv2d_35 (Separa  (None, 36, 36, 64)       4736
 bleConv2D)

 batch_normalization_29 (Bat  (None, 36, 36, 64)       256
 chNormalization)

 max_pooling2d_22 (MaxPoolin  (None, 18, 18, 64)       0
 g2D)

 separable_conv2d_36 (Separa  (None, 18, 18, 128)      8896
 bleConv2D)
```



```
separable_conv2d_37 (Separa  (None, 18, 18, 128)      17664
bleConv2D)

batch_normalization_30 (Bat  (None, 18, 18, 128)      512
chNormalization)

max_pooling2d_23 (MaxPoolin  (None, 9, 9, 128)        0
g2D)

dropout_20 (Dropout)        (None, 9, 9, 128)         0

separable_conv2d_38 (Separa  (None, 9, 9, 256)        34176
bleConv2D)

separable_conv2d_39 (Separa  (None, 9, 9, 256)        68096
bleConv2D)

batch_normalization_31 (Bat  (None, 9, 9, 256)        1024
chNormalization)

max_pooling2d_24 (MaxPoolin  (None, 4, 4, 256)        0
g2D)

dropout_21 (Dropout)        (None, 4, 4, 256)         0

flatten_4 (Flatten)         (None, 4096)              0

dense_16 (Dense)            (None, 512)               2097664

batch_normalization_32 (Bat  (None, 512)              2048
chNormalization)

dropout_22 (Dropout)        (None, 512)               0

dense_17 (Dense)            (None, 128)               65664

batch_normalization_33 (Bat  (None, 128)              512
 dropout_23 (Dropout)        (None, 128)               0

 dense_18 (Dense)            (None, 64)                8256

 batch_normalization_34 (Bat  (None, 64)               256
 chNormalization)

 dropout_24 (Dropout)        (None, 64)                0

 dense_19 (Dense)            (None, 4)                 260

=================================================================
Total params: 2,317,348
Trainable params: 2,314,980
Non-trainable params: 2,368
_________________________________________________________________
```

Model Compilation:

The chosen optimizer is Adam, which is a popular algorithm for gradient-based optimization of neural networks. The chosen loss function is Categorical Crossentropy, which is commonly used in multi-class classification problems. Additionally, the AUC (Area Under the Curve) metric is specified to be used for evaluating the model during training.

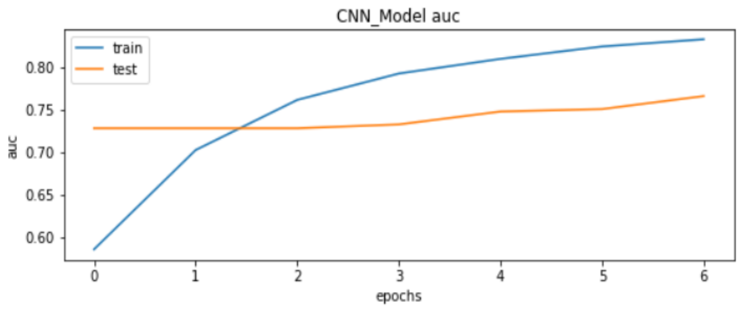
Results:

The results for epochs 7 and 50 show that the model has achieved stability by epoch 50, as evidenced by the AUC graphs. Both epochs 7 and 50 show a reduction in the loss function, indicating that the model is improving over time. However, the AUC graphs suggest that the model's performance has further improved by epoch 50, and it is likely that the loss function has also continued to decrease beyond epoch 7. Overall, the results suggest that the model has improved over time and has achieved better stability and performance by epoch 50 compared to epoch 7.

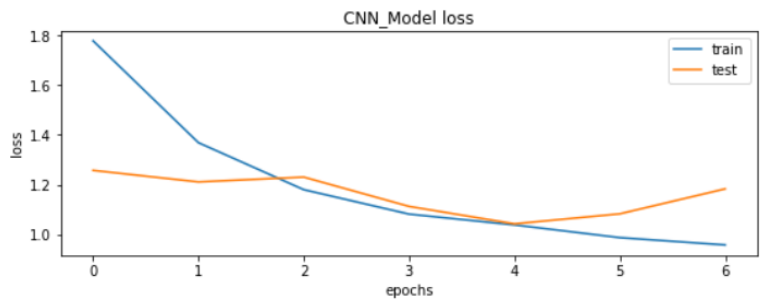
Epoch 7:

```
Epoch 1/7
65/65 [==============================] - 85s 1s/step - loss: 1.6822 - auc: 0.5993 - val_loss: 1.2056 - val_auc: 0.8092
Epoch 2/7
65/65 [==============================] - 82s 1s/step - loss: 1.3308 - auc: 0.7067 - val_loss: 1.1470 - val_auc: 0.7281
Epoch 3/7
65/65 [==============================] - 80s 1s/step - loss: 1.1552 - auc: 0.7699 - val_loss: 1.0694 - val_auc: 0.7281
Epoch 4/7
65/65 [==============================] - 71s 1s/step - loss: 1.0497 - auc: 0.8068 - val_loss: 1.0121 - val_auc: 0.8092
Epoch 5/7
65/65 [==============================] - 68s 1s/step - loss: 1.0074 - auc: 0.8188 - val_loss: 1.0035 - val_auc: 0.8353
Epoch 6/7
65/65 [==============================] - 73s 1s/step - loss: 0.9484 - auc: 0.8382 - val_loss: 1.0264 - val_auc: 0.8442
Epoch 7/7
65/65 [==============================] - 72s 1s/step - loss: 0.8912 - auc: 0.8561 - val_loss: 1.0771 - val_auc: 0.8401
```

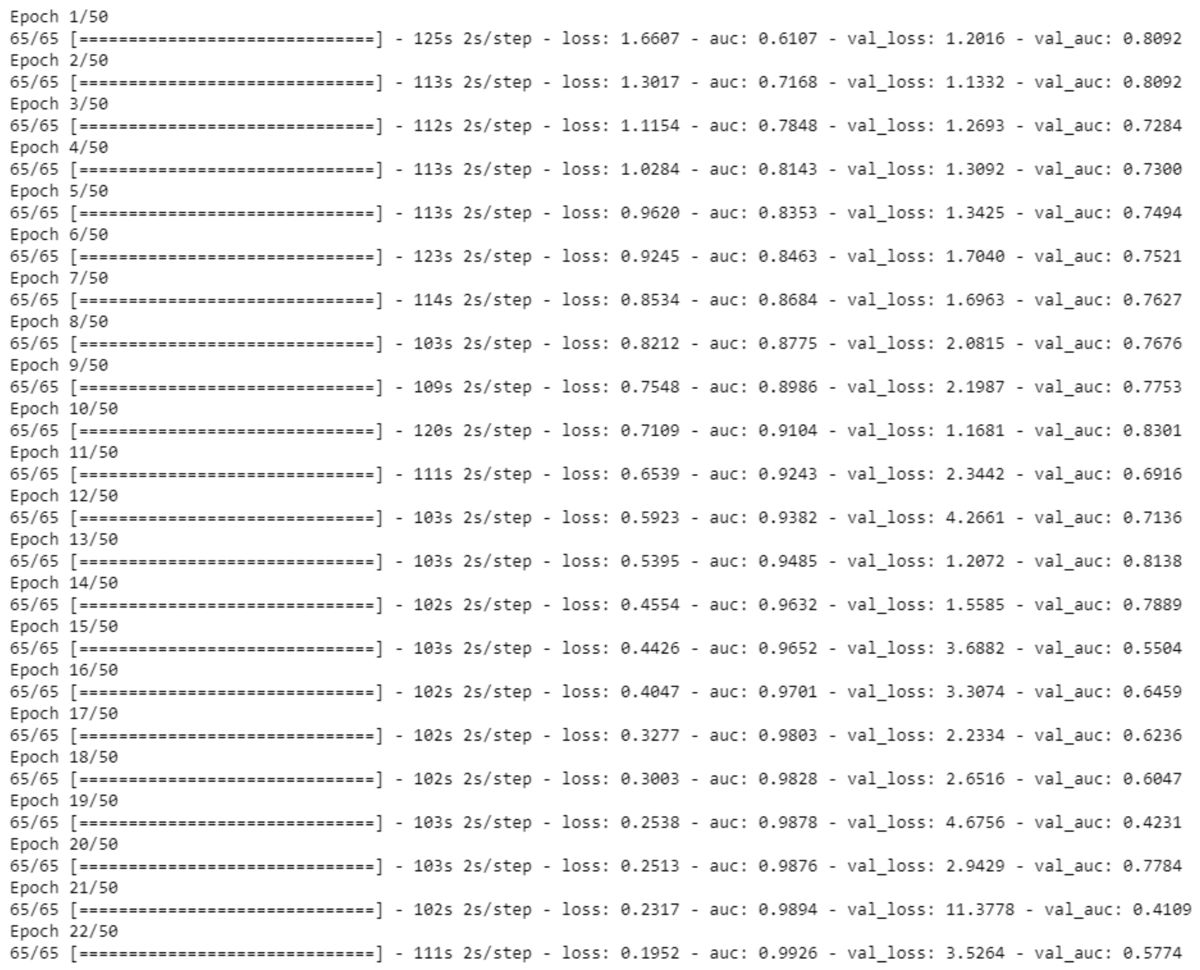
Epoch 7 – AUC graph:



Epoch 7 – Model Loss:



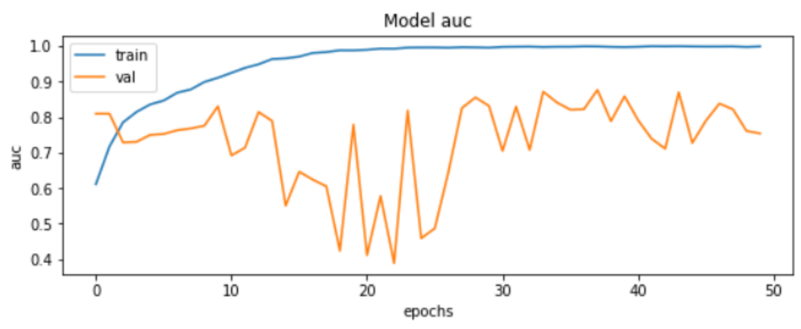
Epoch 50:



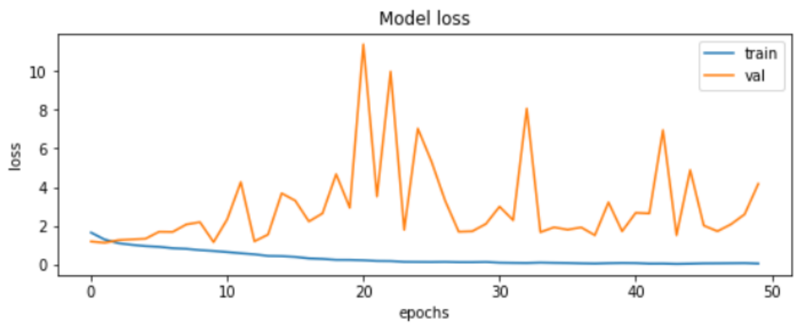
```
Epoch 1/50
65/65 [==============================] - 125s 2s/step - loss: 1.6607 - auc: 0.6107 - val_loss: 1.2016 - val_auc: 0.8092
Epoch 2/50
65/65 [==============================] - 113s 2s/step - loss: 1.3017 - auc: 0.7168 - val_loss: 1.1332 - val_auc: 0.8092
Epoch 3/50
65/65 [==============================] - 112s 2s/step - loss: 1.1154 - auc: 0.7848 - val_loss: 1.2693 - val_auc: 0.7284
Epoch 4/50
65/65 [==============================] - 113s 2s/step - loss: 1.0284 - auc: 0.8143 - val_loss: 1.3092 - val_auc: 0.7300
Epoch 5/50
65/65 [==============================] - 113s 2s/step - loss: 0.9620 - auc: 0.8353 - val_loss: 1.3425 - val_auc: 0.7494
Epoch 6/50
65/65 [==============================] - 123s 2s/step - loss: 0.9245 - auc: 0.8463 - val_loss: 1.7040 - val_auc: 0.7521
Epoch 7/50
65/65 [==============================] - 114s 2s/step - loss: 0.8534 - auc: 0.8684 - val_loss: 1.6963 - val_auc: 0.7627
Epoch 8/50
65/65 [==============================] - 103s 2s/step - loss: 0.8212 - auc: 0.8775 - val_loss: 2.0815 - val_auc: 0.7676
Epoch 9/50
65/65 [==============================] - 109s 2s/step - loss: 0.7548 - auc: 0.8986 - val_loss: 2.1987 - val_auc: 0.7753
Epoch 10/50
65/65 [==============================] - 120s 2s/step - loss: 0.7109 - auc: 0.9104 - val_loss: 1.1681 - val_auc: 0.8301
Epoch 11/50
65/65 [==============================] - 111s 2s/step - loss: 0.6539 - auc: 0.9243 - val_loss: 2.3442 - val_auc: 0.6916
Epoch 12/50
65/65 [==============================] - 103s 2s/step - loss: 0.5923 - auc: 0.9382 - val_loss: 4.2661 - val_auc: 0.7136
Epoch 13/50
65/65 [==============================] - 103s 2s/step - loss: 0.5395 - auc: 0.9485 - val_loss: 1.2072 - val_auc: 0.8138
Epoch 14/50
65/65 [==============================] - 102s 2s/step - loss: 0.4554 - auc: 0.9632 - val_loss: 1.5585 - val_auc: 0.7889
Epoch 15/50
65/65 [==============================] - 103s 2s/step - loss: 0.4426 - auc: 0.9652 - val_loss: 3.6882 - val_auc: 0.5504
Epoch 16/50
65/65 [==============================] - 102s 2s/step - loss: 0.4047 - auc: 0.9701 - val_loss: 3.3074 - val_auc: 0.6459
Epoch 17/50
65/65 [==============================] - 102s 2s/step - loss: 0.3277 - auc: 0.9803 - val_loss: 2.2334 - val_auc: 0.6236
Epoch 18/50
65/65 [==============================] - 102s 2s/step - loss: 0.3003 - auc: 0.9828 - val_loss: 2.6516 - val_auc: 0.6047
Epoch 19/50
65/65 [==============================] - 103s 2s/step - loss: 0.2538 - auc: 0.9878 - val_loss: 4.6756 - val_auc: 0.4231
Epoch 20/50
65/65 [==============================] - 103s 2s/step - loss: 0.2513 - auc: 0.9876 - val_loss: 2.9429 - val_auc: 0.7784
Epoch 21/50
65/65 [==============================] - 102s 2s/step - loss: 0.2317 - auc: 0.9894 - val_loss: 11.3778 - val_auc: 0.4109
Epoch 22/50
65/65 [==============================] - 111s 2s/step - loss: 0.1952 - auc: 0.9926 - val_loss: 3.5264 - val_auc: 0.5774
```

```
Epoch 23/50
65/65 [==============================] - 103s 2s/step - loss: 0.1888 - auc: 0.9924 - val_loss: 9.9733 - val_auc: 0.3885
Epoch 24/50
65/65 [==============================] - 103s 2s/step - loss: 0.1502 - auc: 0.9953 - val_loss: 1.8032 - val_auc: 0.8183
Epoch 25/50
65/65 [==============================] - 102s 2s/step - loss: 0.1445 - auc: 0.9957 - val_loss: 7.0294 - val_auc: 0.4583
Epoch 26/50
65/65 [==============================] - 104s 2s/step - loss: 0.1404 - auc: 0.9957 - val_loss: 5.3332 - val_auc: 0.4863
Epoch 27/50
65/65 [==============================] - 102s 2s/step - loss: 0.1474 - auc: 0.9950 - val_loss: 3.3152 - val_auc: 0.6439
Epoch 28/50
65/65 [==============================] - 106s 2s/step - loss: 0.1305 - auc: 0.9965 - val_loss: 1.7053 - val_auc: 0.8250
Epoch 29/50
65/65 [==============================] - 103s 2s/step - loss: 0.1297 - auc: 0.9960 - val_loss: 1.7288 - val_auc: 0.8552
Epoch 30/50
65/65 [==============================] - 103s 2s/step - loss: 0.1413 - auc: 0.9951 - val_loss: 2.1090 - val_auc: 0.8311
Epoch 31/50
65/65 [==============================] - 103s 2s/step - loss: 0.1061 - auc: 0.9974 - val_loss: 3.0010 - val_auc: 0.7044
Epoch 32/50
65/65 [==============================] - 103s 2s/step - loss: 0.0967 - auc: 0.9979 - val_loss: 2.2998 - val_auc: 0.8293
Epoch 33/50
65/65 [==============================] - 103s 2s/step - loss: 0.0910 - auc: 0.9983 - val_loss: 8.0685 - val_auc: 0.7072
Epoch 34/50
65/65 [==============================] - 103s 2s/step - loss: 0.1109 - auc: 0.9971 - val_loss: 1.6805 - val_auc: 0.8713
Epoch 35/50
65/65 [==============================] - 104s 2s/step - loss: 0.0980 - auc: 0.9978 - val_loss: 1.9328 - val_auc: 0.8411
Epoch 36/50
65/65 [==============================] - 102s 2s/step - loss: 0.0900 - auc: 0.9979 - val_loss: 1.8108 - val_auc: 0.8206
Epoch 37/50
65/65 [==============================] - 103s 2s/step - loss: 0.0741 - auc: 0.9989 - val_loss: 1.9275 - val_auc: 0.8218
Epoch 38/50
65/65 [==============================] - 103s 2s/step - loss: 0.0635 - auc: 0.9987 - val_loss: 1.5228 - val_auc: 0.8763
Epoch 39/50
65/65 [==============================] - 105s 2s/step - loss: 0.0821 - auc: 0.9975 - val_loss: 3.2230 - val_auc: 0.7883

Epoch 40/50
65/65 [==============================] - 106s 2s/step - loss: 0.0924 - auc: 0.9969 - val_loss: 1.7269 - val_auc: 0.8585
Epoch 41/50
65/65 [==============================] - 103s 2s/step - loss: 0.0867 - auc: 0.9978 - val_loss: 2.6813 - val_auc: 0.7910
Epoch 42/50
65/65 [==============================] - 103s 2s/step - loss: 0.0584 - auc: 0.9994 - val_loss: 2.6486 - val_auc: 0.7387
Epoch 43/50
65/65 [==============================] - 104s 2s/step - loss: 0.0611 - auc: 0.9990 - val_loss: 6.9508 - val_auc: 0.7107
Epoch 44/50
65/65 [==============================] - 103s 2s/step - loss: 0.0437 - auc: 0.9994 - val_loss: 1.5268 - val_auc: 0.8696
Epoch 45/50
65/65 [==============================] - 103s 2s/step - loss: 0.0558 - auc: 0.9987 - val_loss: 4.8915 - val_auc: 0.7270
Epoch 46/50
65/65 [==============================] - 106s 2s/step - loss: 0.0719 - auc: 0.9984 - val_loss: 2.0218 - val_auc: 0.7894
Epoch 47/50
65/65 [==============================] - 103s 2s/step - loss: 0.0747 - auc: 0.9985 - val_loss: 1.7287 - val_auc: 0.8378
Epoch 48/50
65/65 [==============================] - 106s 2s/step - loss: 0.0809 - auc: 0.9987 - val_loss: 2.0899 - val_auc: 0.8215
Epoch 49/50
65/65 [==============================] - 103s 2s/step - loss: 0.0864 - auc: 0.9973 - val_loss: 2.6070 - val_auc: 0.7606
Epoch 50/50
65/65 [==============================] - 104s 2s/step - loss: 0.0630 - auc: 0.9987 - val_loss: 4.1771 - val_auc: 0.7536
```

<u>Epoch 50- AUC graph:</u>



Epoch 50 – Loss Function:



Accuracy results:

No. Of Epochs	Train Accuracy	Test Accuracy
7	0.77	0.76
50	0.8983	0.7536

Team Members:

- Hari Priya Avarampalayam Manoharan
- Rida Fathima
- Subramanian Arumugam
- Agash Sekar
- Abinesh