**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.

A screenshot of a computer

Description automatically generated with low confidence

**Sample MRI scan image dataset:**

Shape

Description automatically generated with medium confidence

**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.

A picture containing text, microphone

Description automatically generated

**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 |

**Output of model Summary:**

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

**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:**

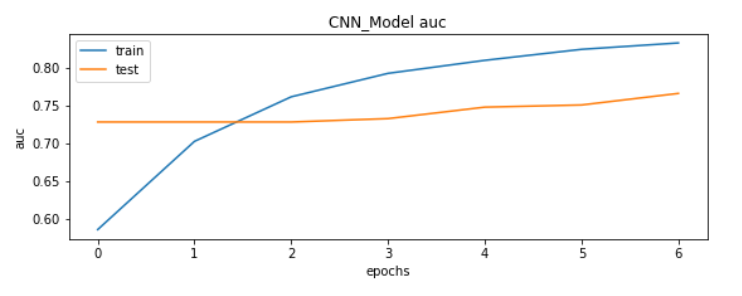
The results for epochs 7 and 50 show that the model has achieved stability by epoch 50, as evidenced by the AUC graphs. Both epochs 7 and 50 show a reduction in the loss function, indicating that the model is improving over time. However, the AUC graphs suggest that the model's performance has further 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:**

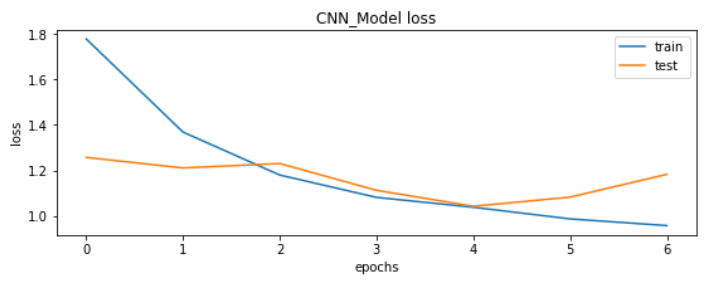
A picture containing text

Description automatically generated

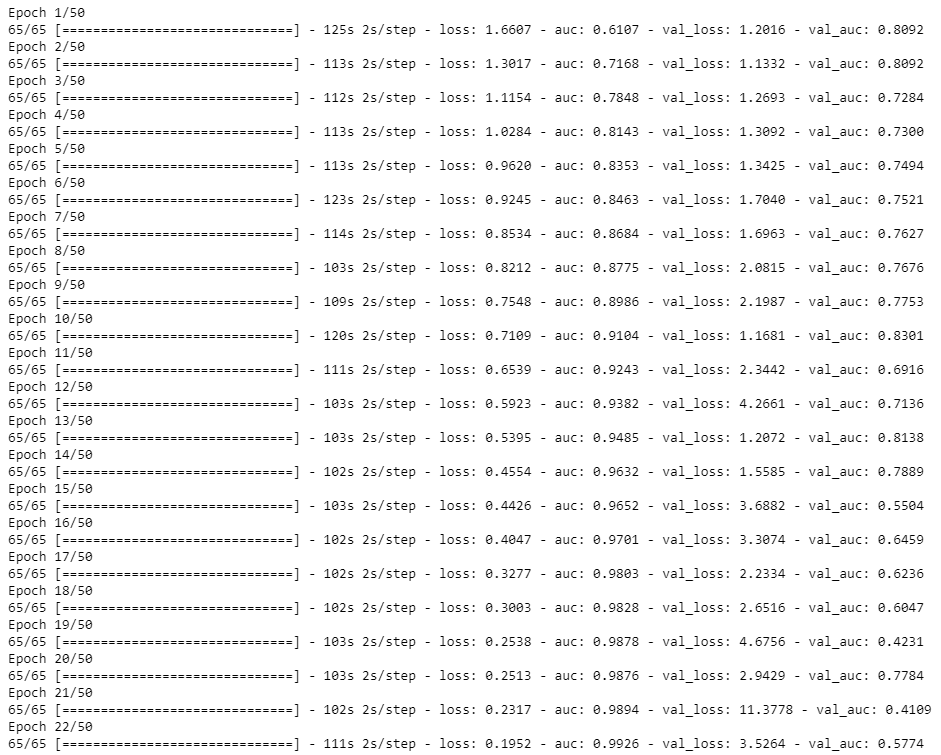
**Epoch 7 – AUC graph:**

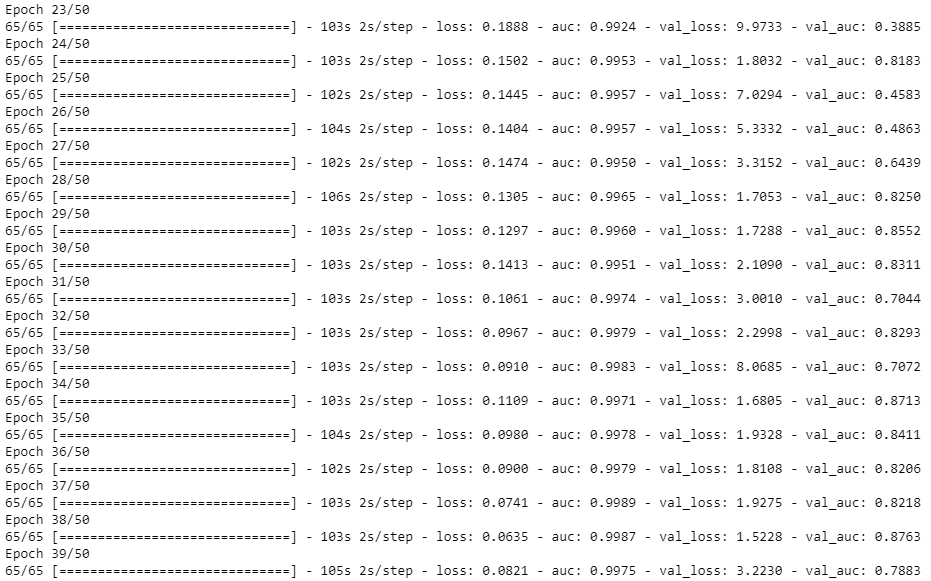


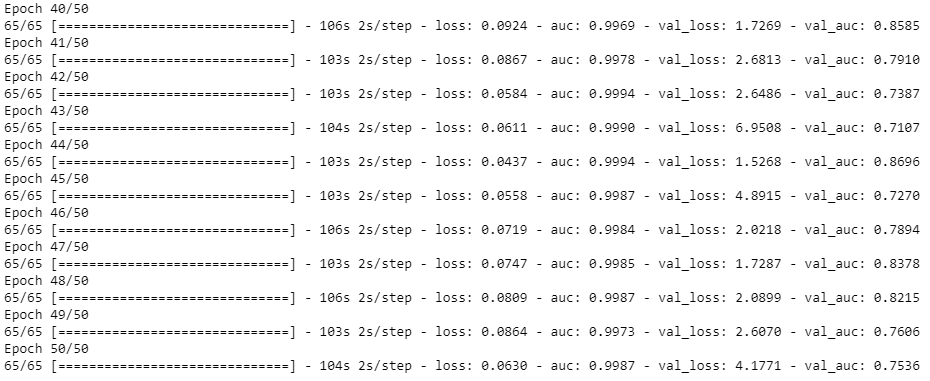
**Epoch 7 – Model Loss:**



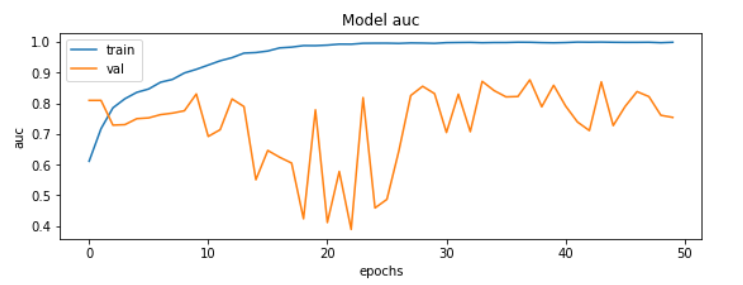
**Epoch 50:**



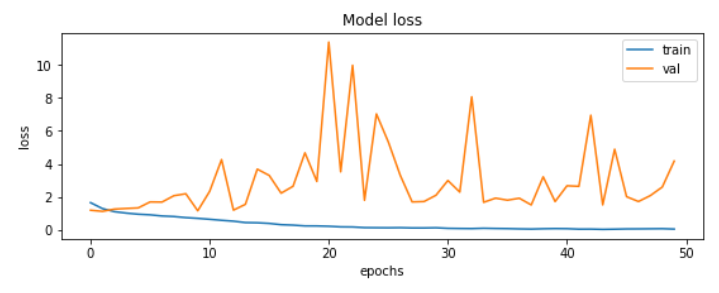




**Epoch 50- AUC graph:**



**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