METHODOLOGY

This chapter describes the project's methodologies with the aid of a block diagram and a flow chart. This chapter outlines the procedures for developing a CNN model to forecast Alzheimer's disease. This chapter explains how a user-friendly Python application is created to identify Alzheimer's disease.

1 Deep Learning Model Building

1.1 Dataset

MRI Alzheimer preprocessed dataset is downloaded from Kaggle website. There are four classes of images in the dataset. There are 6400 MRI images in the entire dataset. Mild Demented, Class 1 (896 images), Moderate Demented, Class 2 (64 images), Non-Demented Class 3 (3200 images) and Very Mild Demented Class 4 (2240 images). This dataset has imbalanced data where Moderate Demented class has only 64 images. Each class include two-dimensional coronal slices of medial temporal lobe which is fetched from three-dimensional MRI image. Data in this dataset, are collected from elderly people from different areas.

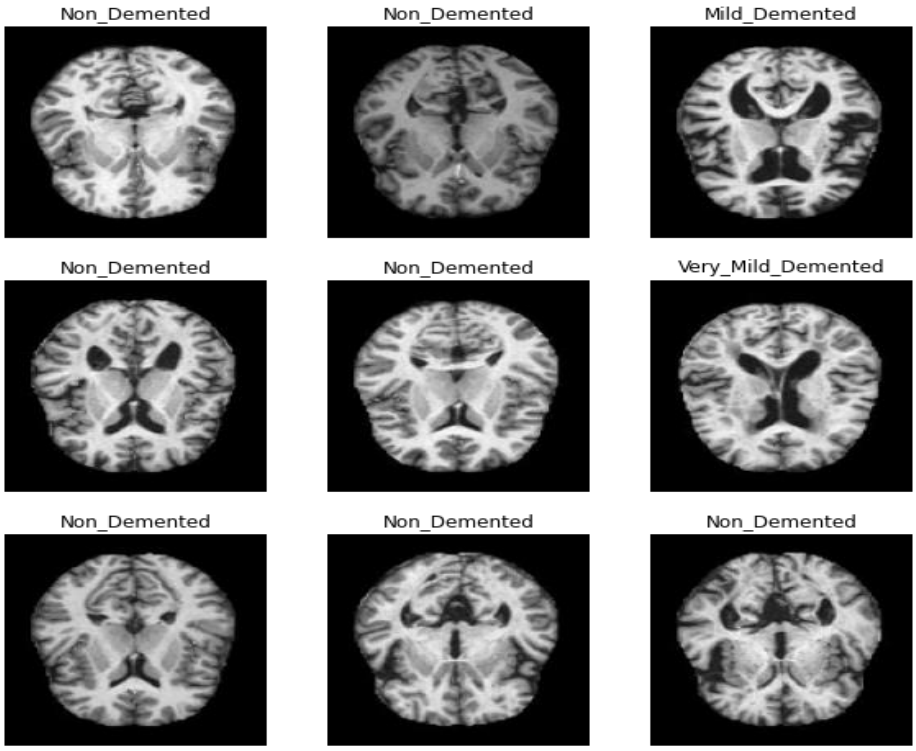

Fig 1 Dataset

1.2 Preprocessing

Augmentation technique is used to increase the number of images in moderate demented class because that class causes the dataset is imbalanced. Different image

augmentation techniques are used to the images in the input directory. Some of the techniques used in this code are edge enhancement, histogram equalization, contrast enhancement, and additive Gaussian noise. The augmentation is carried out using the imgaug package, which provides a reliable and flexible technique for using image augmentations.

Create training, validation, and testing sets using a collection of magnetic resonance imaging (MRI) data for Alzheimer's disease. This process is frequently used to get image data ready for deep learning models. The original dataset is divided into three directories with a defined ratio using the splitfolders package. In this instance, the training set receives 80% of the photos, while the validation and testing sets each receive 10%. All of the photos in the collection have been normalised and scaled to 128 * 128.

1.3 Feature Study

With the aid of clinical indicators and imaging data, researchers have been developing classifiers to aid in the diagnosis of AD. These investigations have revealed significant structural variations between the AD brain and the healthy brain in areas like the entorhinal cortex and hippocampal entorhinal cortex. One of the first brain areas to be impacted by AD is the entorhinal cortex, which is in charge of storing and retrieving memories. According to studies, people with AD have a smaller entorhinal cortex volume, and imaging methods like magnetic resonance imaging can be used to detect this reduction. (MRI). Similarly, the hippocampus is a region of the brain that is involved in memory formation and spatial navigation. Studies have also shown that the volume of the hippocampus is reduced in individuals with AD, particularly in the early stages of the disease.

1.4 Deep Learning Model

The Keras API in TensorFlow is used to construct the Sequencial model. Rescaling is the initial layer that is added to the model, scaling the input image's pixel values to be between 0 and 1. Conv2D and MaxPooling2D are the following two layers, and they are used to extract features from the input images. The Conv2D layers apply the ReLU activation function and convolutions to the input picture using a predetermined number of filters and kernel sizes. The MaxPooling2D layers take the maximum value within a certain pooling window to minimise the spatial dimensionality of the feature maps. With the padding option set to "same," padding is added to the input image to make it the same size as the input image. The second and third convolutional layers, respectively, are followed by the addition of two dropout layers. In order to avoid overfitting, these layers randomly remove a certain number of connections between the layers during training. The Flatten layer is used to flatten the feature maps into a 1D vector after the convolutional layers. The final step is the addition of two fully connected dense layers, each with a particular number of units and activation function. The final Dense layer employs the softmax activation function to generate a probability distribution over the classes and contains 4 units, which is equal to the number of classes in the dataset.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 128, 128, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 128, 128, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2097280
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 260
=================================================================
```

Fig 2 2D CNN Layer

This model has three convolutional layers, two dropout layers, and two fully connected layers overall, making it a reasonably straightforward CNN. The activation functions and initialization approaches employed in the model are standard procedures in deep learning for image classification, and it is intended to categorize images into one of four classes.

1.5 Model Training and Evaluation

Using Keras' fit technique, train the specified CNN model. The fit method iterates over the training dataset in batches, training the model for a predetermined number of epochs. In this instance, the validation dataset is supplied as the validation_data parameter, while the training dataset is supplied as the first argument. The number of times the complete training dataset is run through the model during training is determined by the epochs parameter. The model is trained for 100 epochs in this instance. The batch_size option determines how many samples the model will process at once. In this instance, the model runs 64 batches of samples. Evaluates the trained model on the test dataset using the evaluate method in Keras. The evaluate method computes the loss and accuracy of the model on the test dataset.

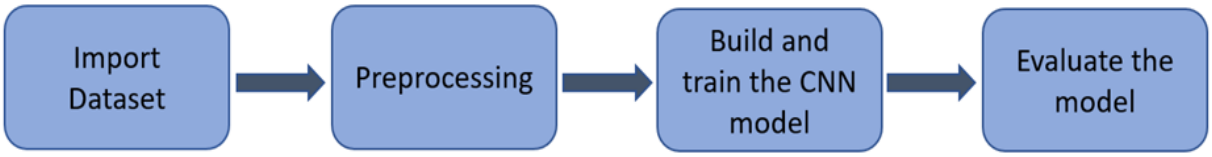

Fig 4.3 Block Diagram

2 Python Application

Kivy is a Python package that allows to create multi-touch applications on different platforms. The application loads NifTi file. The NifTi file is internally transformed into a 2D image of coronal slices and several image processing methods are used. The Medial Temporal Lobe's range is manually inputted. A trained model will forecast each and every image according to the specified range, average out all the forecasts, and then provide the final forecast.

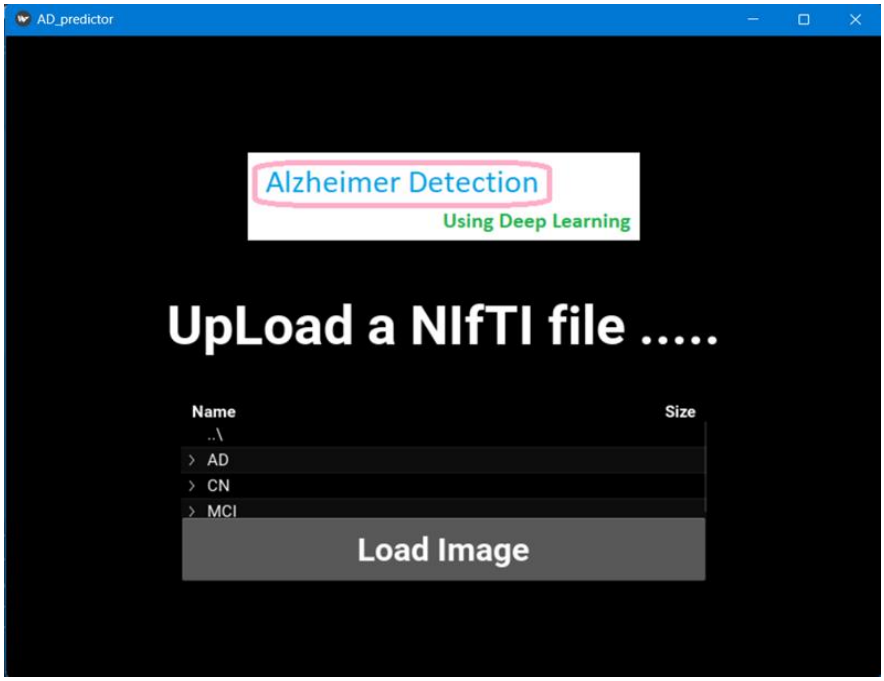

Fig 4 App view

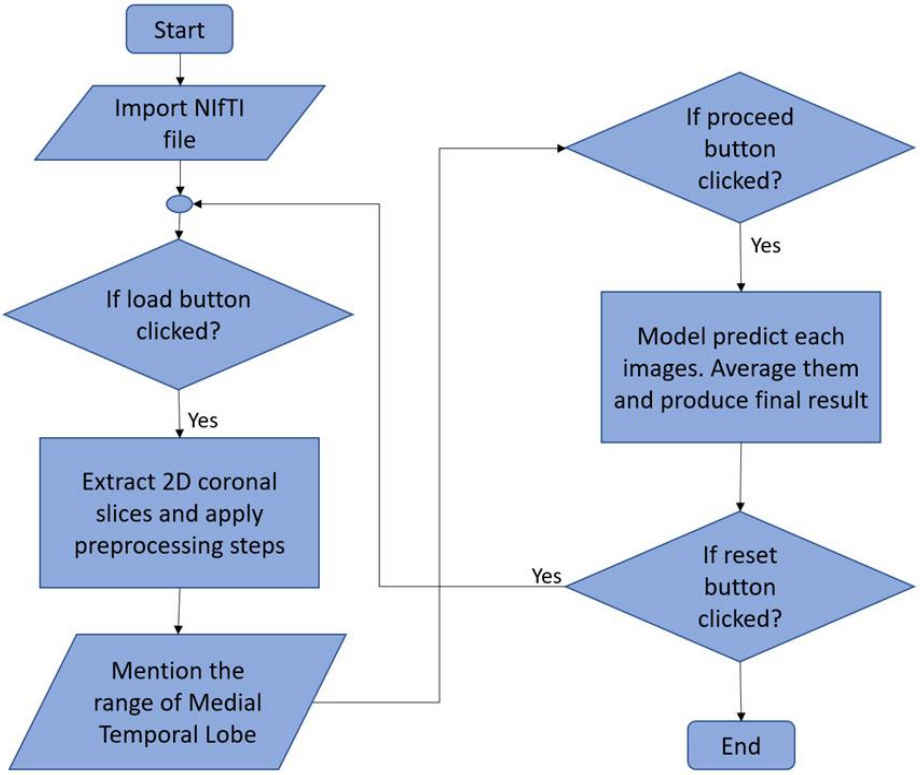

Fig 5 Flowchart

Widgets for the application's label, logo, input, and buttons are added. Load the NIfTI file using the listview widget. The 3D mri image (NIfTI file) is extracted into 2D coronal slices when the load button is pressed. Few image processing techniques are applied . The range of the medial temporal region should be provided by the user to the input widget. The model receives each image individually after clicking the proceed button. Each image is predicted by the model, which then averages the values to give the final output that is displayed by the label widget.

RESULT AND DISCUSSION

This chapter deals with the results obtained by evaluation of the model and python application.

1 Model Evaluation

Utilises Keras package evaluate method to assess the trained model against the test dataset. On the test dataset, the evaluate method calculates the model's loss and accuracy. The augmentation is performed using the imgaug library, which provides a powerful and flexible way to apply image augmentations. Every image in the input directory is iterated over by the code, which then adds a series of augmentations to each one. The final photos can be added to the original dataset to help the deep learning model that was built on it perform better.

```
[ ]  loss, accuracy = model.evaluate(test_ds)

     6/6 [==============================] - 8s 1s/step - loss: 0.0550 - accuracy: 0.9826
```

Fig 6 Before augmentation

```
[ ]  loss, accuracy = model.evaluate(test_ds)

     11/11 [==============================] - 0s 12ms/step - loss: 0.0091 - accuracy: 0.9953
```

Fig 7 After augmentation

Overall, the model's excellent performance on the test dataset shows that it can correctly categorise photos of healthy and Alzheimer's brains with a high level of precision.

2 Visualization

A deep learning model's performance during training can be seen in the training data accuracy and loss graph. The number of epochs, or the number of times the model has been trained on the complete dataset, is plotted against the accuracy and loss graphs. The accuracy and loss values should, in theory, converge to an optimal value as the number of epochs rises.

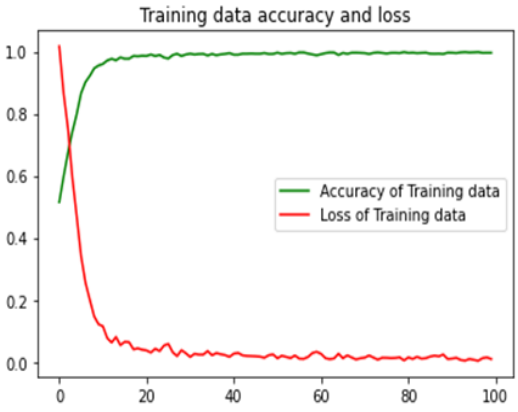

Fig 8 Training data accuracy and lose

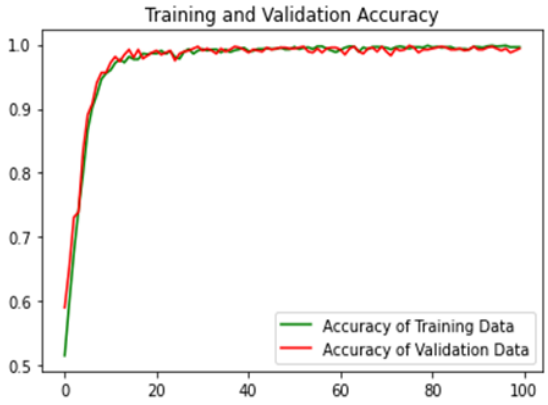

Fig 9 Training and validation accuracy

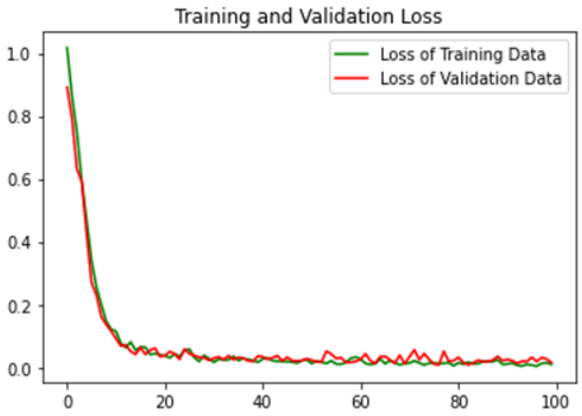

Fig 10 Training and validation loss

In Training and validation loss graph, both the training and validation loss decrease and plateau together, it indicates that the model is learning well and is not overfitting. In training and validation accuracy graph, both the training and validation accuracy increase and plateau together, it also indicates that the model is trained well and not overfitted.

The training accuracy and validation accuracy are commonly plotted on a graph during the training process, with the accuracy on the y-axis and the quantity of training iterations (or epochs) on the x-axis. The validation accuracy should finally plateau while the training accuracy should continue to rise with each passing epoch. The training and validation loss graph shows the evolution of the training process's loss values over time. The validation loss indicates the error on the validation data, which is data that the model has not seen during training, while the training loss shows the error on the training data during the training process.

3 Confusion Matrix

A table called a confusion matrix is frequently used to assess how well a classification model is working. It displays the proportion of accurate and inaccurate predictions the model made in relation to the actual results (or ground truth) in a specific dataset. The confusion matrix for a 4-class classification model will contain 4 rows and 4 columns.

While the columns indicate the classes that the model predicted, the rows represent the actual classes of the data.

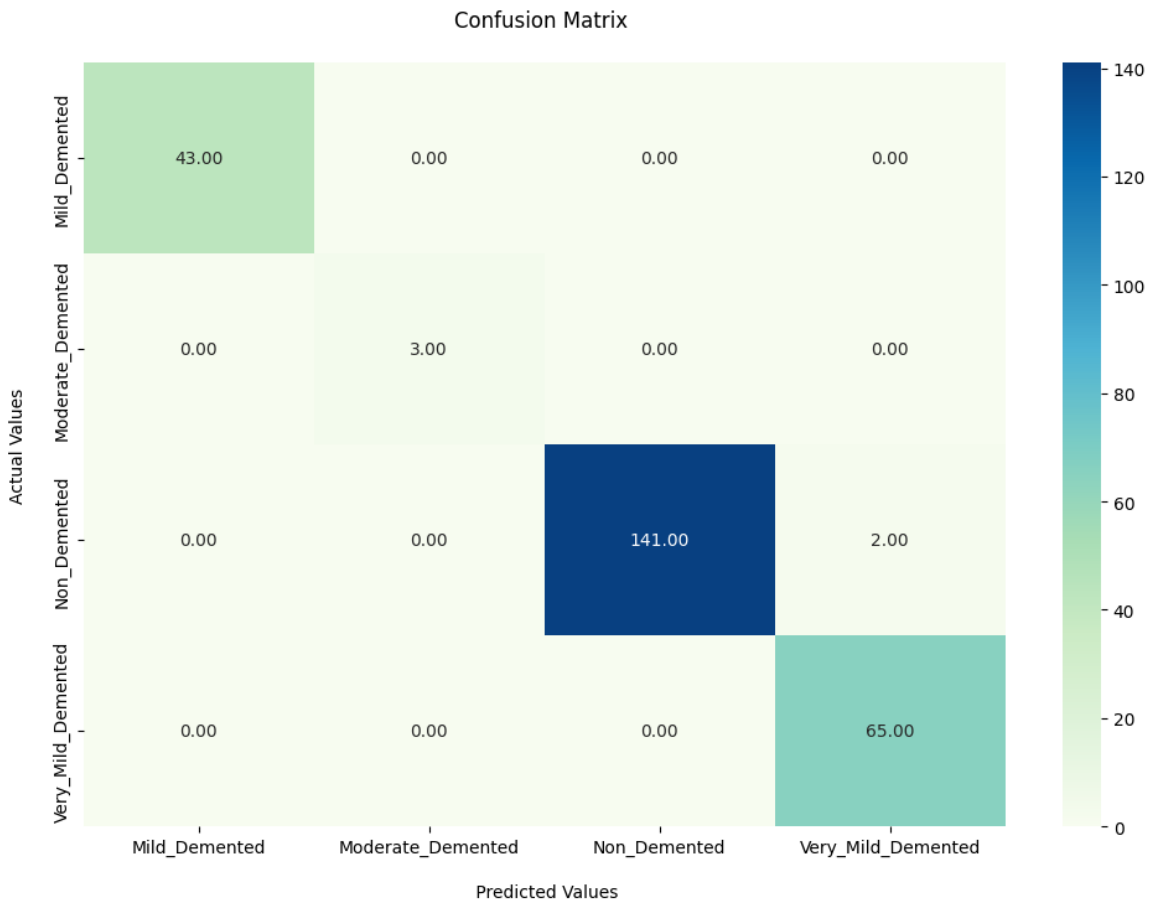

Fig 11 Confusion Matrix

The true positive (TP) values, or the number of cases that were correctly identified as belonging to the relevant class, are represented by the diagonal members of the matrix. The false positive (FP), false negative (FN), and true negative (TN) values are represented by the off-diagonal elements. The performance of the classification model can be evaluated by computing evaluation metrics like accuracy, precision, recall, and F1 score using the TP, FP, FN, and TN data.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	43
1	1.00	1.00	1.00	3
2	1.00	0.99	0.99	143
3	0.97	1.00	0.98	65
accuracy			0.99	254
macro avg	0.99	1.00	0.99	254
weighted avg	0.99	0.99	0.99	254

Fig 12 Classification Report

The performance metrics of the classification model will be shown in the classification report. For each class, it will show metrics like precision, recall, f1-score, and support. Additionally, data like accuracy, macro average, and weighted average are shown.

4 Python Application Evaluation

The Alzheimer's disease diagnosis using input images from the MRI is predicted by a Python programme for a Kivy-based GUI application. The programme lets users submit NIfTI image files, which are then processed to extract coronal slices, each of which is preprocessed and fed into a deep learning model that has already been trained to produce a diagnosis.

The app has the following features:

- Allows the user to upload a NIfTI file (a medical imaging file format).
- Extracts coronal slices from the uploaded MRI scan and saves them as separate images.
- Allows the user to enter a range of coronal slices to use for prediction.
- Applies contrast stretching and Gaussian blur to each coronal slice image.
- Uses a pre-trained deep learning model to make predictions on each coronal slice.
- Returns the most frequent class prediction across all coronal slices and displays it in the GUI.

The Alzheimer's Disease Neuroimaging Initiative (ADNI) provides a large collection of Magnetic Resonance Imaging (MRI) datasets through its official website. The ADNI MRI dataset is freely available to researchers and clinicians who register with the ADNI website and agree to comply with the data usage policies. Labeled dataset with 3 classes such as AD, MCI (mild) and CN (normal), is downloaded for evaluation of python application.

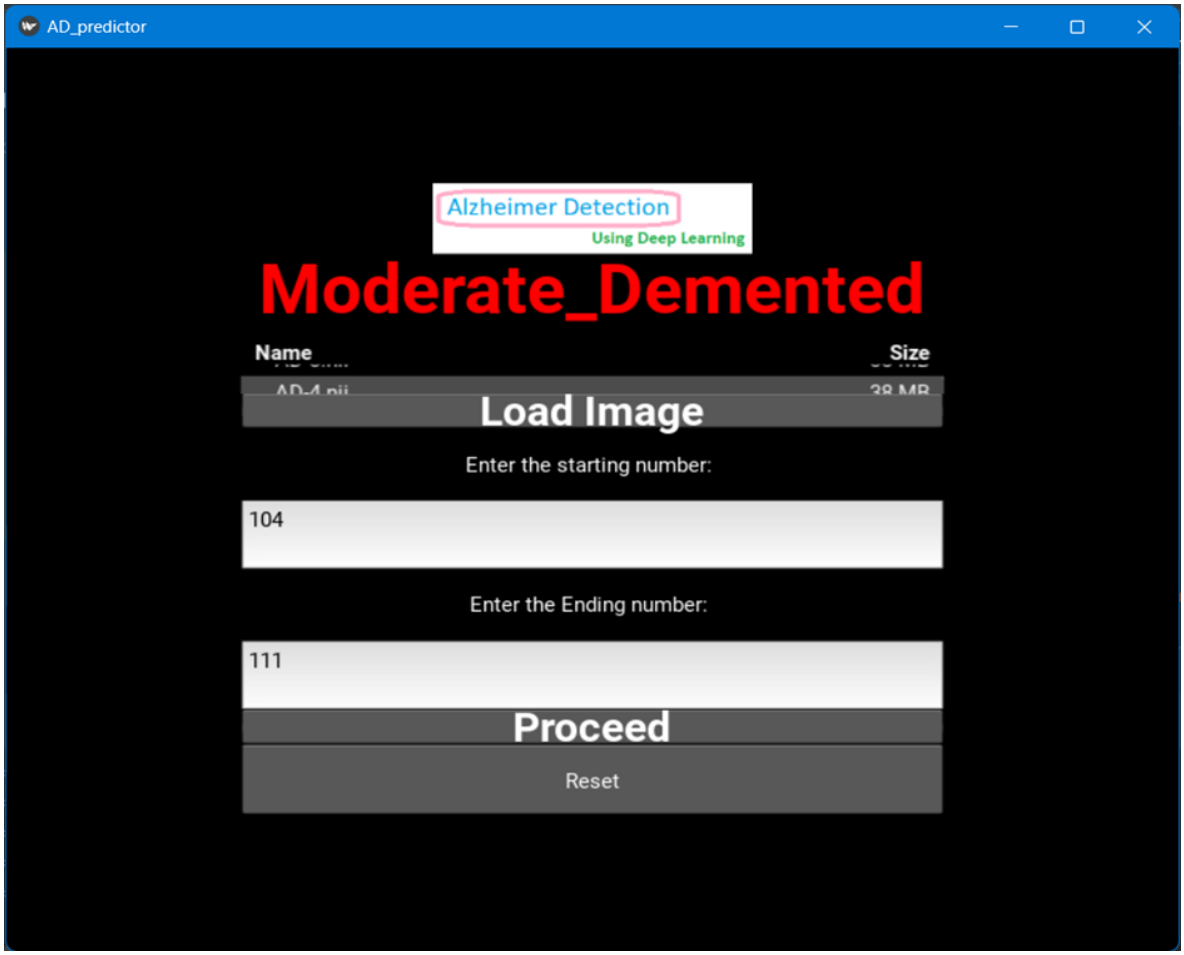

Fig 13 Python application

The Python application shown in the above image allows users to upload NIfTI file (3D MRI images) and enter medial temporal lobe ranges. The label widget displays the final output after the user clicks the proceed button.