Alzheimer's Disease Classification using Transfer Model

Submitted by Team 5:

1-Mohaned Emad

2-Merna Abd El-Moez

3-Rana Hany

4-Rana Ibrahim

Submitted to:

Dr :Eman Ayman

➢ Introduction

Alzheimer's disease, a progressive neurodegenerative disorder, represents one of the most prevalent forms of dementia, affecting millions of individuals worldwide. Characterized by cognitive decline, memory loss, and impaired daily functioning, Alzheimer's poses substantial challenges to affected individuals, their families, and the healthcare system.

Early diagnosis of Alzheimer's disease is crucial for several reasons. First and foremost, it enables timely intervention, allowing healthcare professionals to explore potential treatment options and provide support to patients and their families. Additionally, early detection facilitates the inclusion of individuals in clinical trials, contributing to the development of effective therapies.

In recent years, advancements in medical imaging, particularly magnetic resonance imaging (MRI), have opened new avenues for understanding and diagnosing neurodegenerative diseases. The detailed structural information provided by brain MRI scans offers a valuable resource for developing computational models aimed at classifying Alzheimer's disease stages.

1. Problem Definition and Objective:

Problem: Classifying Alzheimer's disease from brain MRI images.

Objective: Develop a deep learning model for the accurate classification of Alzheimer's disease stages based on brain MRI scans to assist in early diagnosis.

This project seeks to harness the power of deep learning to create a robust model capable of distinguishing between different stages of Alzheimer's disease from MRI scans, contributing to advancements in early detection and management of this debilitating condition.

2. Literature Review:

- **Paper 1**

Title: "Alzheimer Disease Classification through Transfer Learning Approach"

Objective: To classify the stages of Alzheimer's disease using a deep learning model, specifically DenseNet, with transfer learning.

Methodology: Segmented MRI scans into gray matter (GM), white matter (WM), and cerebrospinal fluid (CSF) using SPM12. Utilized 2D GM slices for training and testing. Applied a pre-trained DenseNet model with retraining of the last two blocks.

Results: Achieved a promising accuracy of 97.84% in the multi-class classification of Alzheimer's disease.

Link: Alzheimer Disease Classification through Transfer Learning Approach

- **Paper 2**

Title: "Transfer Learning with DenseNet for Multi-class Classification of Alzheimer's Disease Stages"

Objective: To develop a healthcare decision support system for the multi-class classification of Alzheimer's disease (AD) stages using a transfer learning approach with DenseNet.

Methodology: Employed DenseNet121, DenseNet169, and DenseNet201 for multi-class classification of AD stages (ND, MID, VMD). Adequate preprocessing of MRI images was performed. The model was tested on various datasets, achieving accuracy scores of 92.48%, 93.00%, and 96.05% for DenseNet121, DenseNet169, and DenseNet201, respectively. DenseNet201 demonstrated a remarkable AUC of 0.9901.

Results: The transfer learning architecture, trained with MRI data, exhibited high accuracy in classifying AD stages, showcasing the effectiveness of data augmentation. The proposed DenseNet model outperformed two other variants significantly.

Link :An Alzheimer's disease classification model using transfer learning Densenet with embedded healthcare decision support system

3. Dataset Description and Visualization:

Dataset: Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset

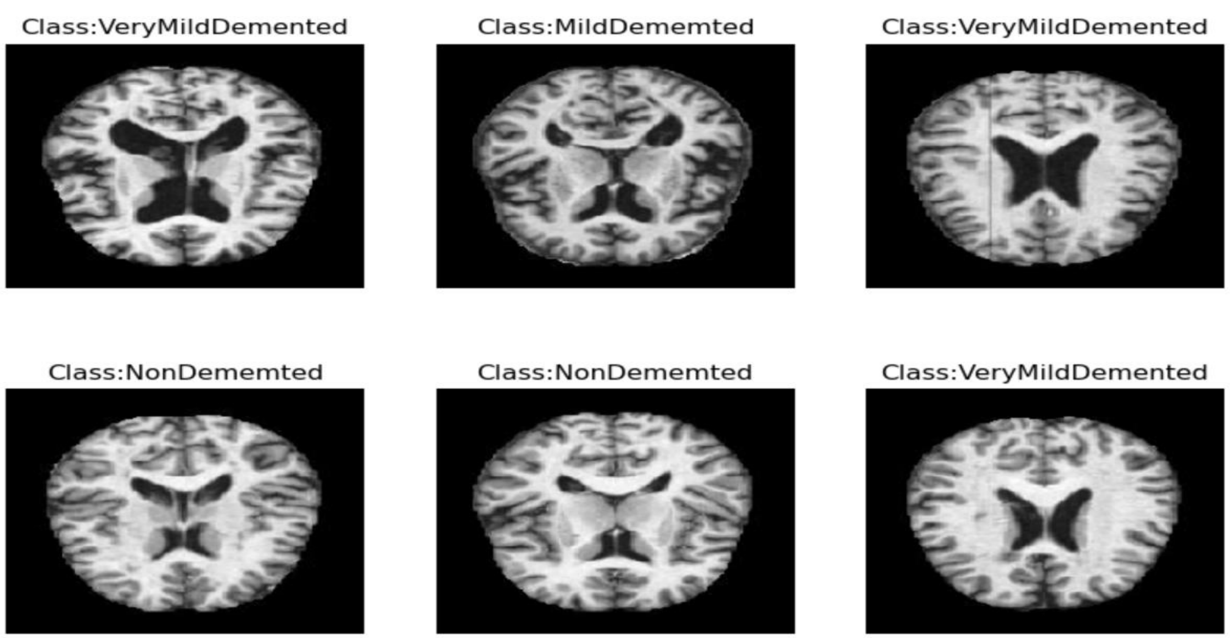
Description: ADNI is a well-established dataset in Alzheimer's disease research. It comprises MRI scans from individuals at different stages of Alzheimer's disease, including labels for stages such as Normal (ND), Mild Cognitive Impairment (MID), and Very Mild Dementia (VMD).

Visualization: To provide a glimpse of the dataset, we include sample images representing different stages of Alzheimer's disease. These images showcase the structural differences observed in MRI scans among individuals with varying cognitive impairment levels.

Link:[alzheimers-dataset-4-class-of-images]

4. Data Preparation:

- Load and preprocess 3D MRI images.
  - scale images
  - resize images.
  - set seed for reproducibility



- Split the dataset into training and testing sets with ratio 80% for training and 20% for testing.

4. Base Model (Inception):

The decision to utilize pretrained models stems from the inherent advantages they offer. By leveraging knowledge gained from a large and diverse dataset related to a different but analogous task, the pretrained 3D model provides a head start. In this project, a model pretrained on the ADNI dataset was chosen due to its comprehensive understanding of spatiotemporal features relevant to Alzheimer's disease.

- **Model Adaptation**:

To adapt the pretrained model for our specific task of multi-class classification for different stages of Alzheimer's disease, we make the following modifications:

- freeze the weights in the base model

-add a single dense layer for classification

-add callbacks for early stopping based on validation AUC

-use AUC as the performance metric

- **Remove Original Output Layer**:

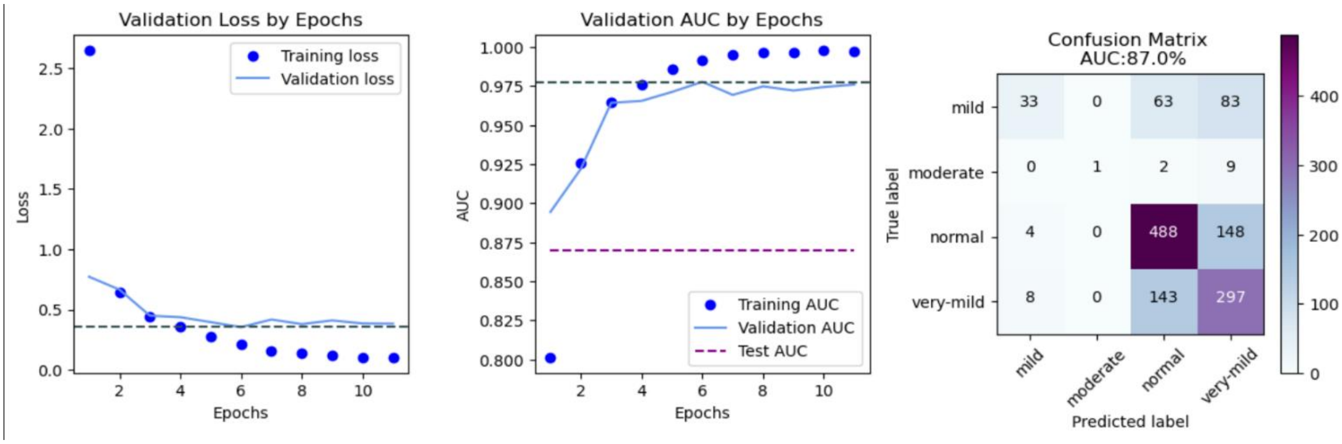
The initial step involved removing the original output layer that was initially designed for the source task of the pretrained model. This is crucial as the source task and the target task (Alzheimer's disease classification) may have different output requirements.

- **Add New Output Layer**:

Following the removal of the original output layer, a new output layer was introduced. This new layer was meticulously designed to have the appropriate number of neurons, aligning with the classes in our Alzheimer's disease classification task. In this scenario, the classes include 'Normal,' 'Mild Cognitive Impairment,' and 'Very Mild Dementia.'

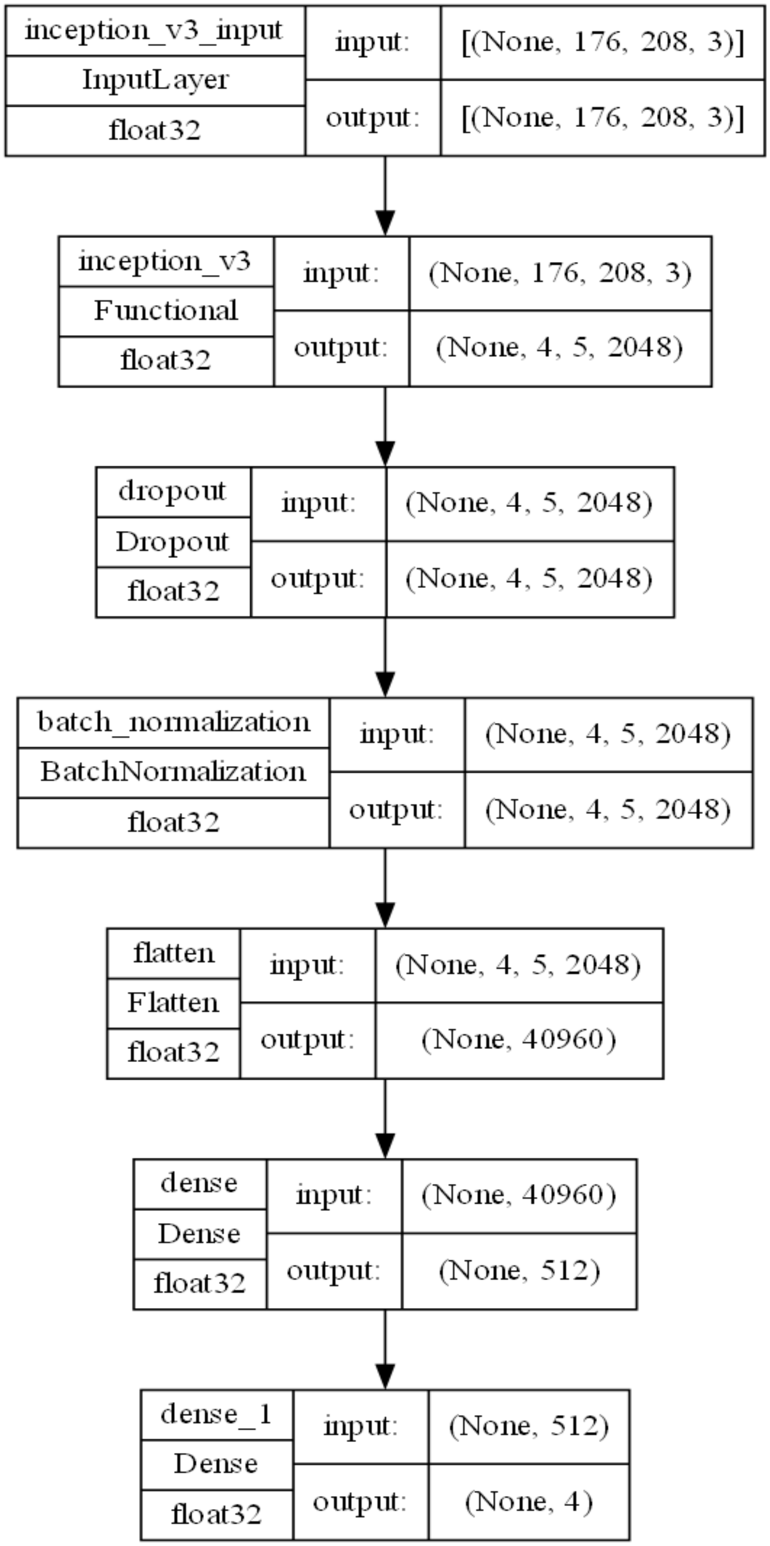
- **Activation Function**:

Apply SoftMax, to obtain class probabilities.



- **Accuracy**: 64%

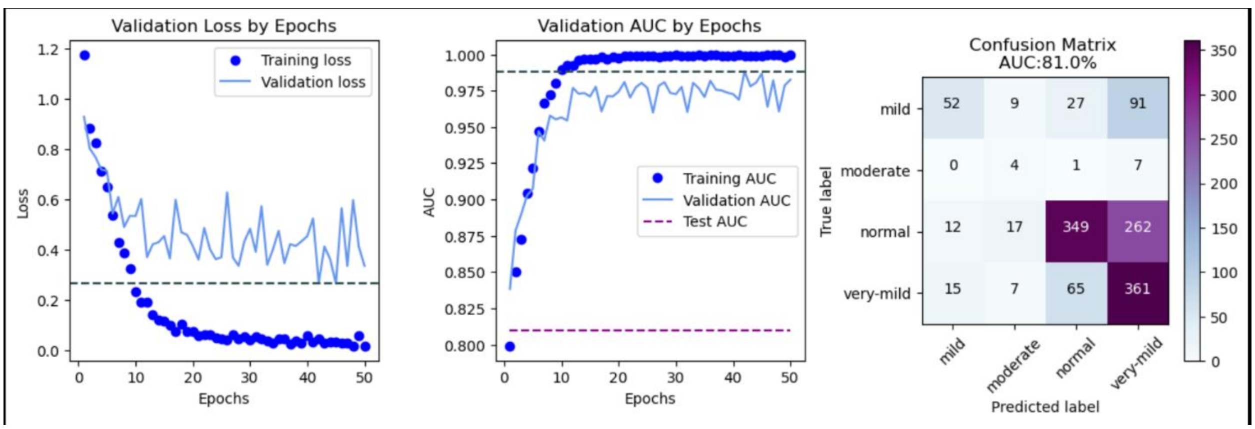
- **Schematic Diagram:**



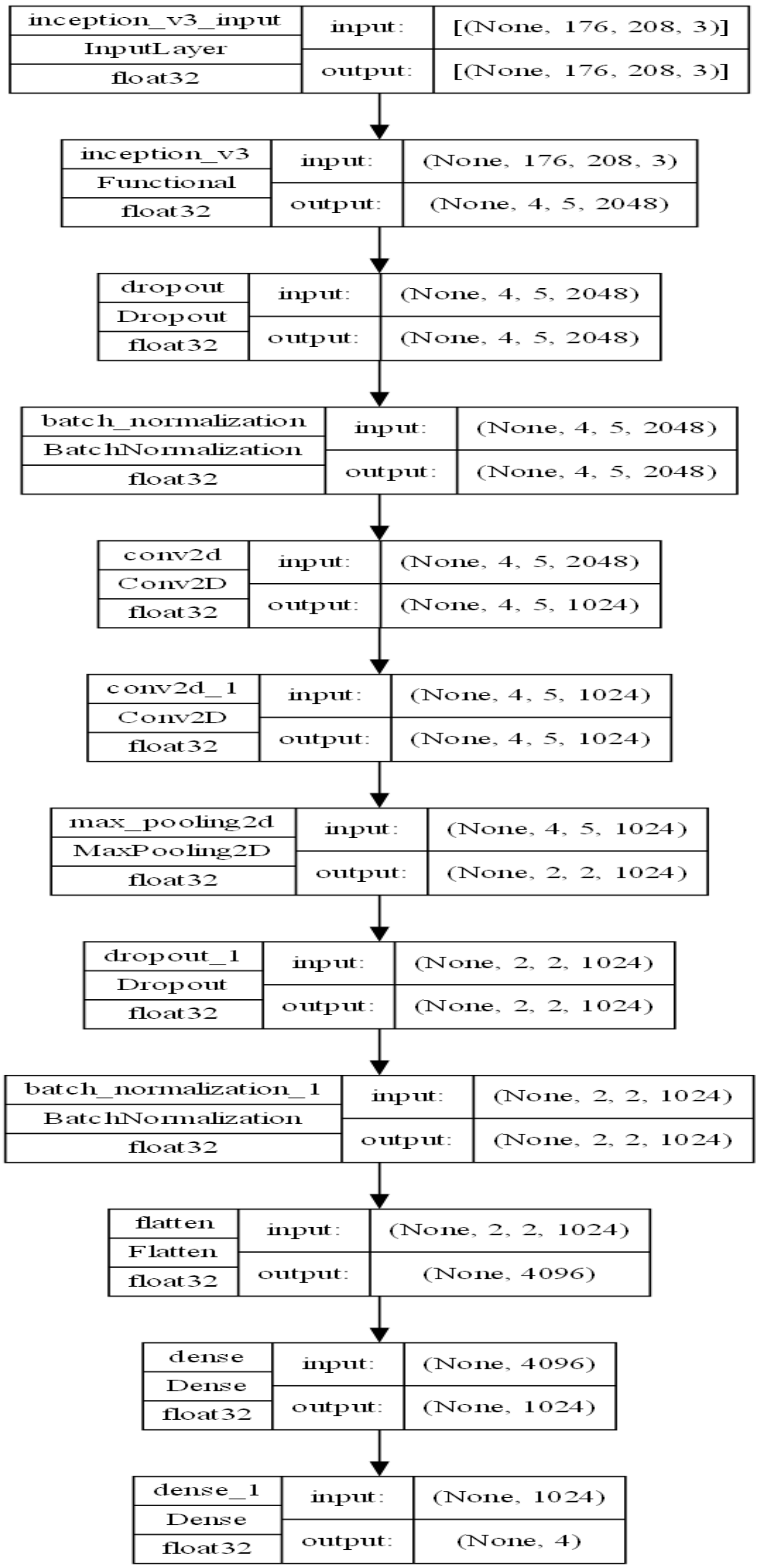
5. Model 1 (Base Model + Additional Capacity):

To increase the capacity of the model additional convolution layers were added along with dropout and batch normalization for regularization. The build transfer model function was adjusted to add the additional layers.

- InceptionV3 pre-trained on ImageNet and remove the top layer
- 'imagenet'-> weights that were learned from training on the ImageNet dataset.
- trainable_layers -> 'none'
- learn_rate=0.001
- EPOCHS = 50
- Added layers
- Optimizer = 'Adam'
- loss = 'categorical_crossentropy'
- Accuracy = 60%



- **Schematic Diagram:**

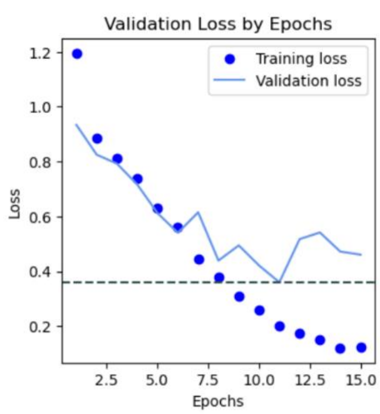


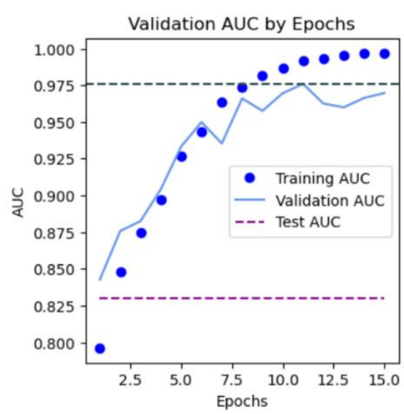
inception_v3_input
InputLayer
float32
input:
[(None, 176, 208, 3)]
output:
[(None, 176, 208, 3)]
inception_v3
Functional
float32
input:
(None, 176, 208, 3)
output:
(None, 4, 5, 2048)
dropout
Dropout
float32
input:
(None, 4, 5, 2048)
output:
(None, 4, 5, 2048)
batch_normalization
BatchNormalization
float32
input:
(None, 4, 5, 2048)
output:
(None, 4, 5, 2048)
conv2d
Conv2D
float32
input:
(None, 4, 5, 2048)
output:
(None, 4, 5, 1024)
conv2d_1
Conv2D
float32
input:
(None, 4, 5, 1024)
output:
(None, 4, 5, 1024)
max_pooling2d
MaxPooling2D
float32
input:
(None, 4, 5, 1024)
output:
(None, 2, 2, 1024)
dropout_1
Dropout
float32
input:
(None, 2, 2, 1024)
output:
(None, 2, 2, 1024)
batch_normalization_1
BatchNormalization
float32
input:
(None, 2, 2, 1024)
output:
(None, 2, 2, 1024)
flatten
Flatten
float32
input:
(None, 2, 2, 1024)
output:
(None, 4096)
dense
Dense
float32
input:
(None, 4096)
output:
(None, 1024)
dense_1
Dense
float32
input:
(None, 1024)
output:
(None, 4)

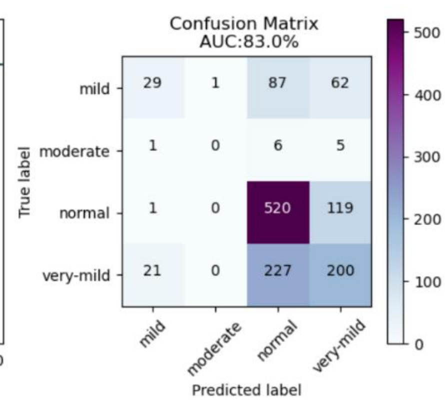
6. Model 2 (Base Model + Optimized Parameters + Data Augmentation):

The model was re-trained with the optimized hyperparameters using data augmentation. The training was slowed down using a reduced schedule for learn-rate, early stopping was removed and the epochs were set to 150 to give the model as much time to train as possible without overfitting

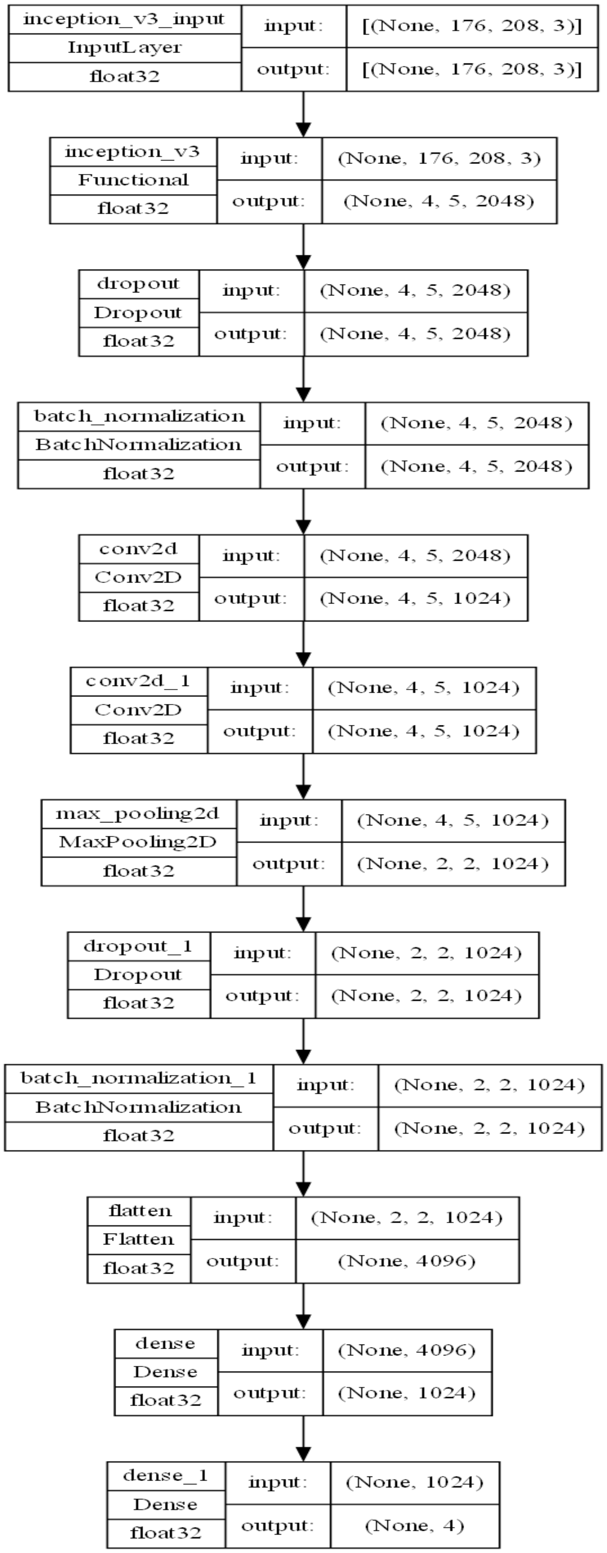
- InceptionV3 pre-trained on ImageNet and remove the top layer
- 'imagenet'-> weights that were learned from training on the ImageNet dataset.
- trainable_layers -> 'none'
- learn_rate=0.001
- EPOCHS = 15
- Added layers
- Optimizer = 'Adam'
- loss = 'categorical_crossentropy'
- Accuracy = 60%







- **Schematic Diagram:**

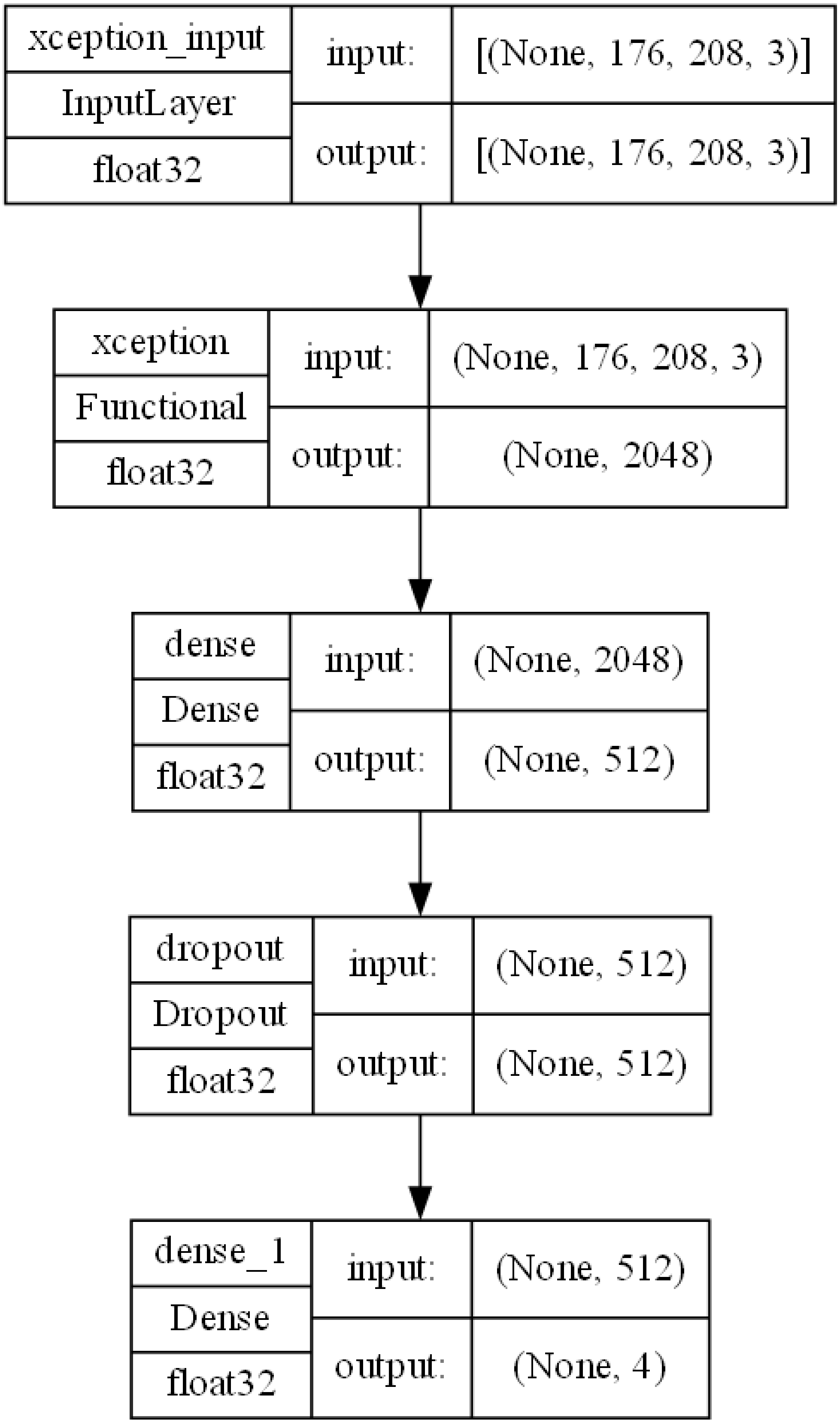


7. Model 3 (Xception):

- Xception pre-trained on ImageNet and remove the top layer
- 'imagenet'-> weights that were learned from training on the ImageNet dataset.
- trainable_layers -> False
- learn_rate=0.001
- EPOCHS = 50
- pooling = 'max', Dense, Dropout, Dense
- Optimizer = 'Adam'
- loss = 'categorical_crossentropy'
- Accuracy = 65%

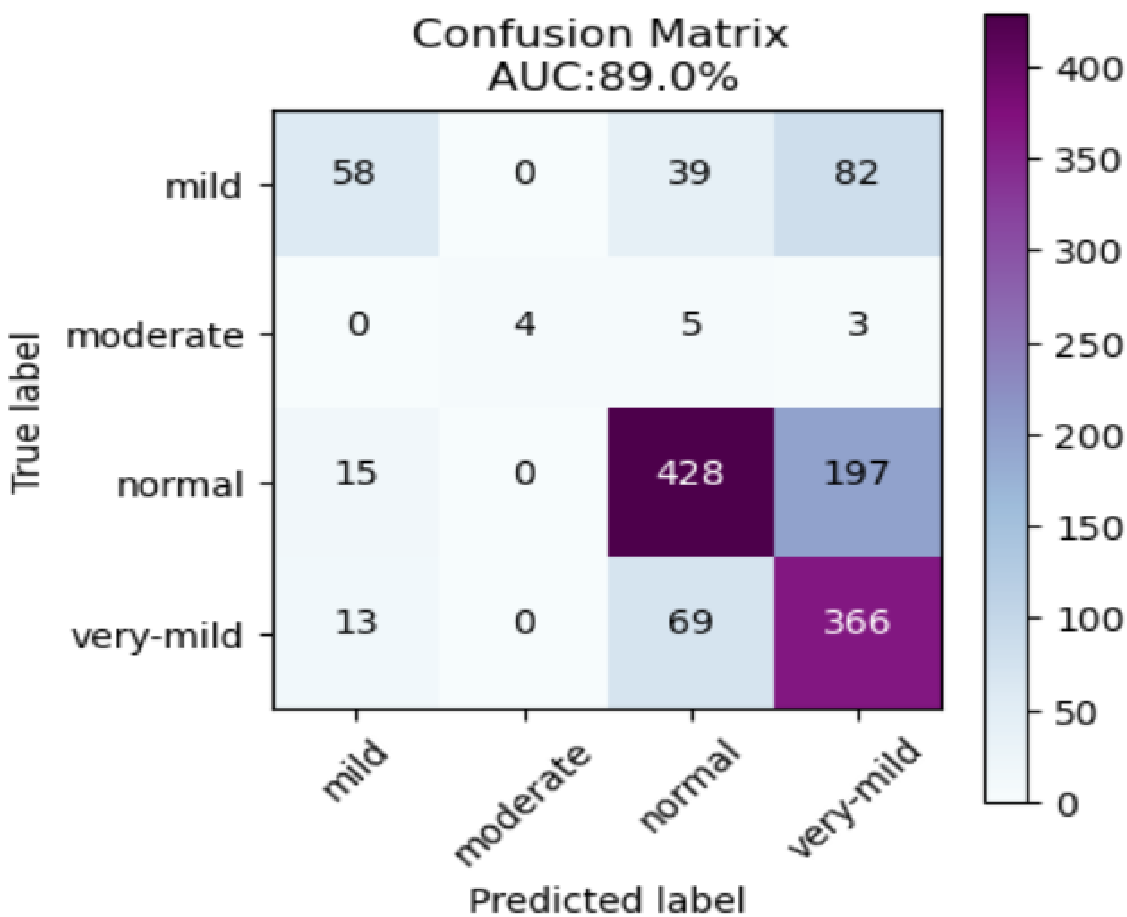
	precision	recall	f1-score	support
MildDememted	0.68	0.43	0.53	179
ModerateDemented	1.00	0.42	0.59	12
NonDememted	0.86	0.59	0.70	640
VeryMildDemented	0.52	0.84	0.64	448
accuracy			0.65	1279
macro avg	0.77	0.57	0.61	1279
weighted avg	0.72	0.65	0.65	1279

- **Schematic Diagram:**

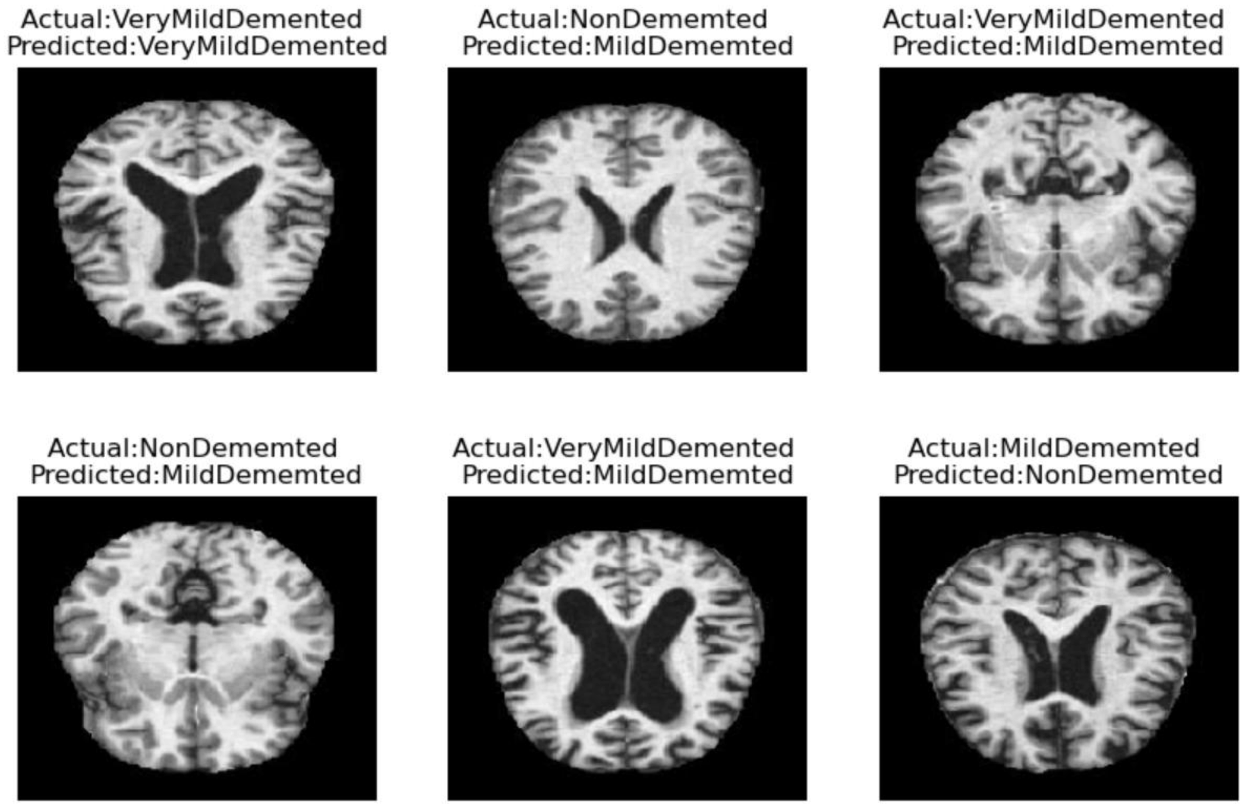


8. Model 4 (VGG19):

- VGG19 pre-trained on ImageNet and remove the top layer
- 'imagenet'-> weights that were learned from training on the ImageNet dataset.
- trainable_layers -> False
- learn_rate=0.001
- EPOCHS = 10
- Flatten and Added Dense layer
- Optimizer = 'Adam'
- loss = 'categorical_crossentropy'
- Accuracy = 67%

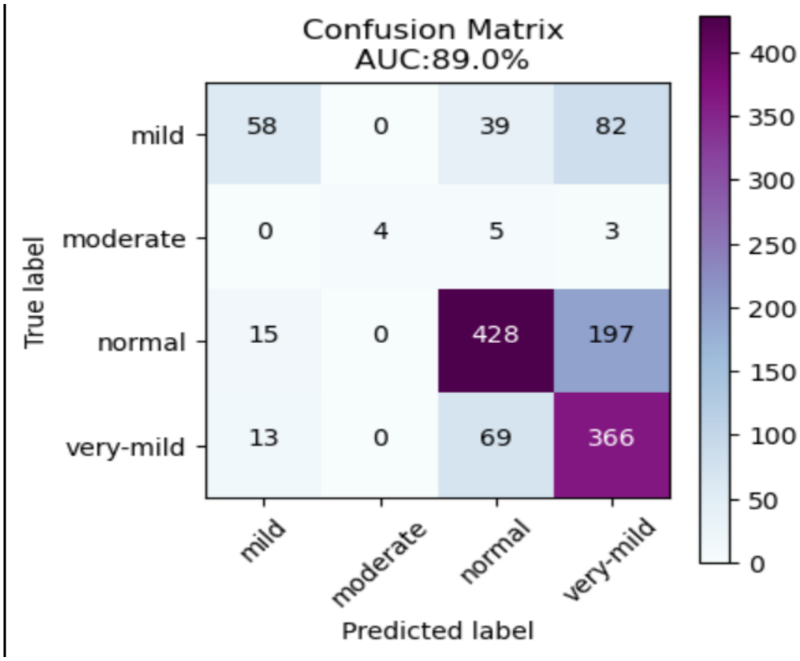


- **View Images and Labels For VGG19**

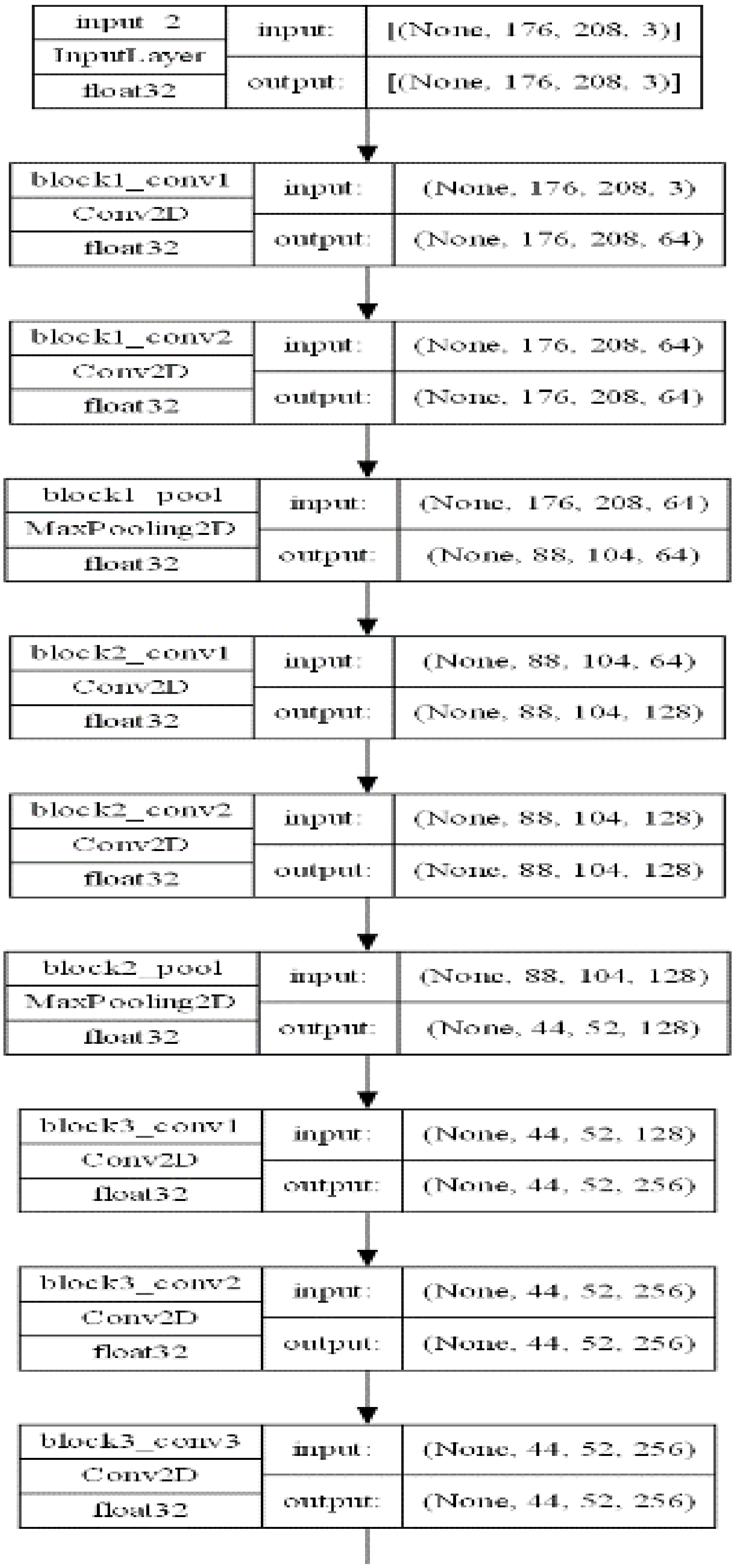


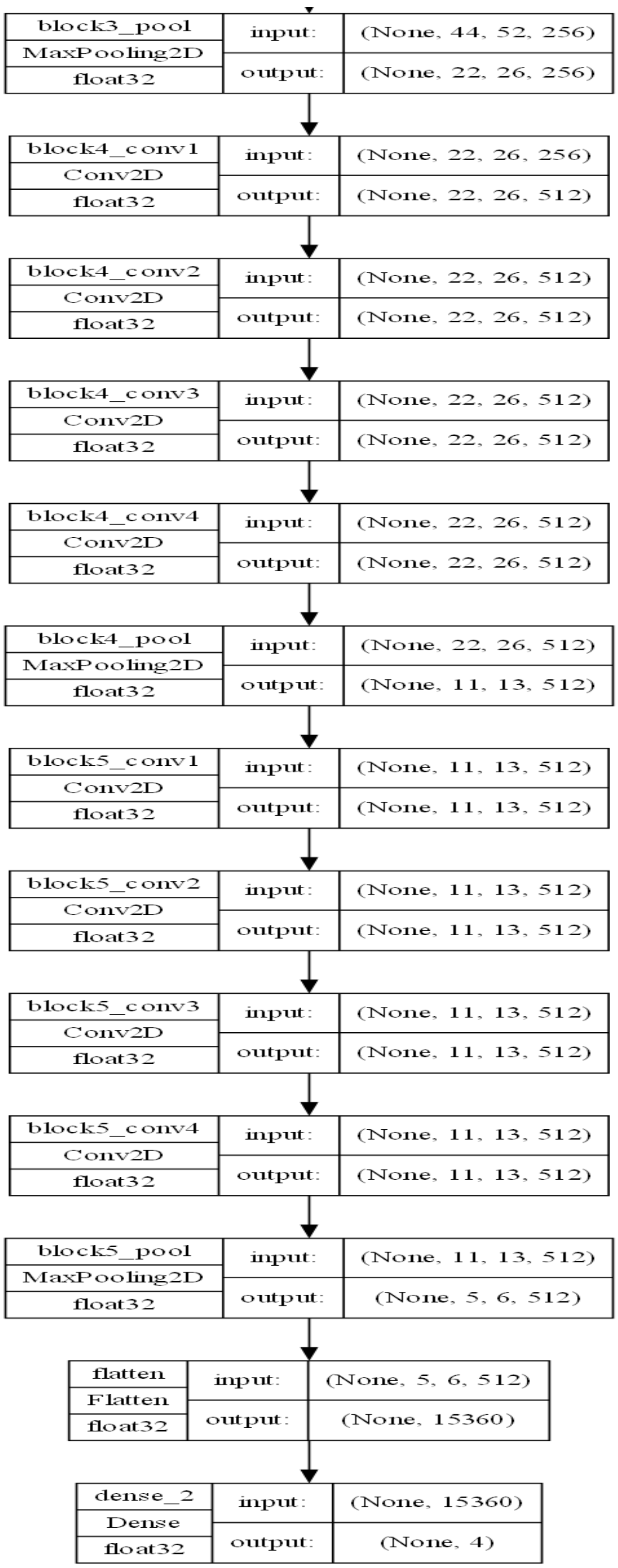
9. The Best Model was Model 4(VGG19):

- **The accuracy was 67%**
- **The confusion matrix**



- **The Schematic Diagram:**





block3_pool
MaxPooling2D
float32
input:
(None, 44, 52, 256)
output:
(None, 22, 26, 256)
block4_conv1
Conv2D
float32
input:
(None, 22, 26, 256)
output:
(None, 22, 26, 512)
block4_conv2
Conv2D
float32
input:
(None, 22, 26, 512)
output:
(None, 22, 26, 512)
block4_conv3
Conv2D
float32
input:
(None, 22, 26, 512)
output:
(None, 22, 26, 512)
block4_conv4
Conv2D
float32
input:
(None, 22, 26, 512)
output:
(None, 22, 26, 512)
block4_pool
MaxPooling2D
float32
input:
(None, 22, 26, 512)
output:
(None, 11, 13, 512)
block5_conv1
Conv2D
float32
input:
(None, 11, 13, 512)
output:
(None, 11, 13, 512)
block5_conv2
Conv2D
float32
input:
(None, 11, 13, 512)
output:
(None, 11, 13, 512)
block5_conv3
Conv2D
float32
input:
(None, 11, 13, 512)
output:
(None, 11, 13, 512)
block5_conv4
Conv2D
float32
input:
(None, 11, 13, 512)
output:
(None, 11, 13, 512)
block5_pool
MaxPooling2D
float32
input:
(None, 11, 13, 512)
output:
(None, 5, 6, 512)
flatten
Flatten
float32
input:
(None, 5, 6, 512)
output:
(None, 15360)
dense_2
Dense
float32
input:
(None, 15360)
output:
(None, 4)

10. The Block diagram: