21BIO211 Intelligence of Biological Systems

ALZHEIMERS DETECTION

Using CNN and Transfer Learning

Presented by Group 9

Group Members

MUHAMMED SHAJAHAN	NIRANJAN PRASANTH	AKSHAY KRISHNAN	KIRAN P
AM.EN.U4AIE21144	AM.EN.U4AIE21148	AM.EN.U4AIE21109	AM.EN.U4AIE21175

ADITHYA VARDHAN REDDY	JAGATH SURYA
AM.EN.U4AIE21171	AM.EN.U4AIE21152

What is Alzheimers

Alzheimer’s disease is a brain disorder that slowly destroys memory and thinking skills and, eventually, the ability to carry out the simplest tasks. Alzheimer’s disease is the most common cause of dementia among older adults.

The disease is named after Dr. Alois Alzheimer. In 1906, Dr. Alzheimer noticed changes in the brain tissue of a woman who had died of an unusual mental illness. Her symptoms included memory loss, language problems, and unpredictable behavior. After she died, he examined her brain and found many abnormal clumps (now called amyloid plaques) and tangled bundles of fibers (now called neurofibrillary, or tau, tangles).

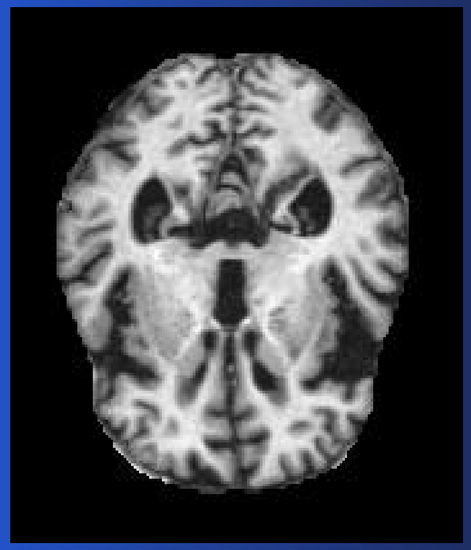
Early detection and accurate classification of Alzheimer's stages play a crucial role in providing appropriate medical interventions and care

Datasets

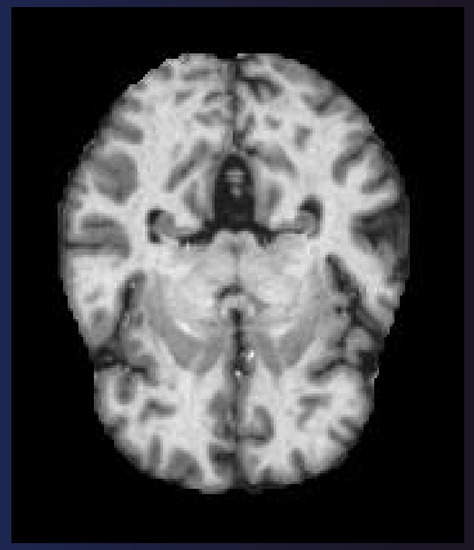
We have collected 2 dataset from Kaggle. The Datasets consists of Preprocessed MRI (Magnetic Resonance Imaging) images of the brain. Each Dataset consists of total 6400 MRI images each.

The data has four classes of images which shows the four stages of dementia:

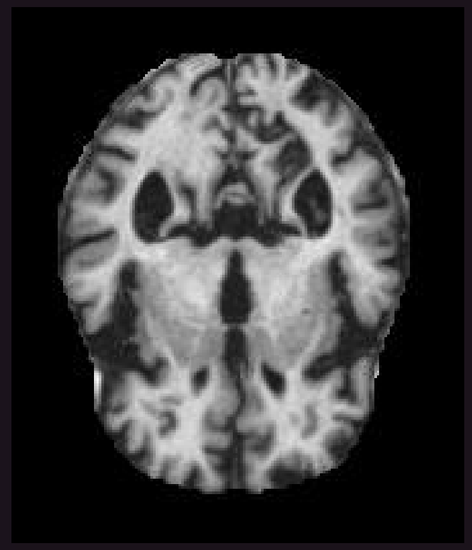
- Mild Demented
- Moderate Demented
- Non Demented
- Very Mild Demented



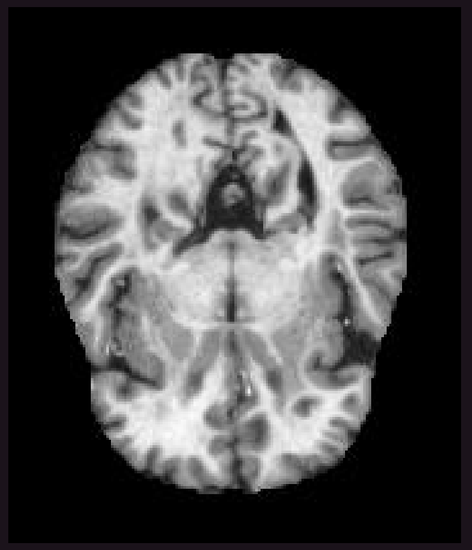
Mild Demented



Very Mild Demented

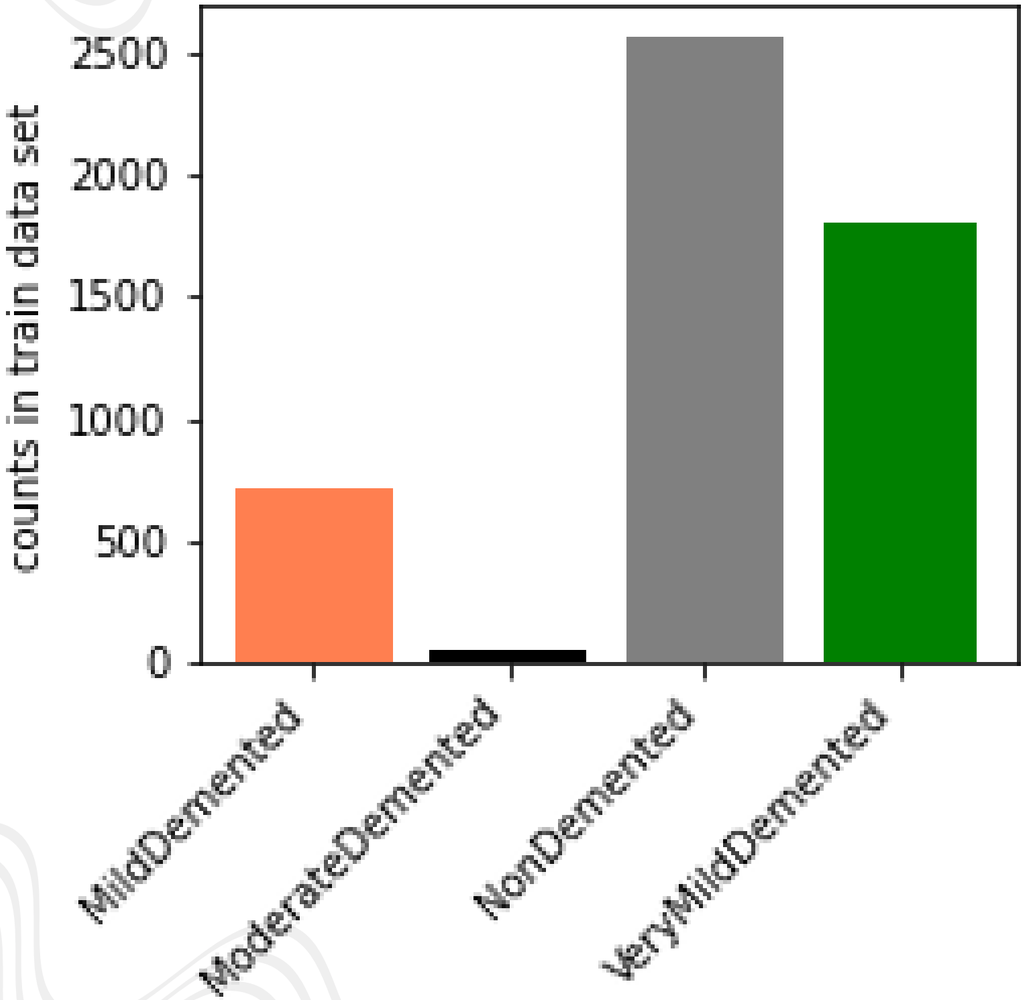


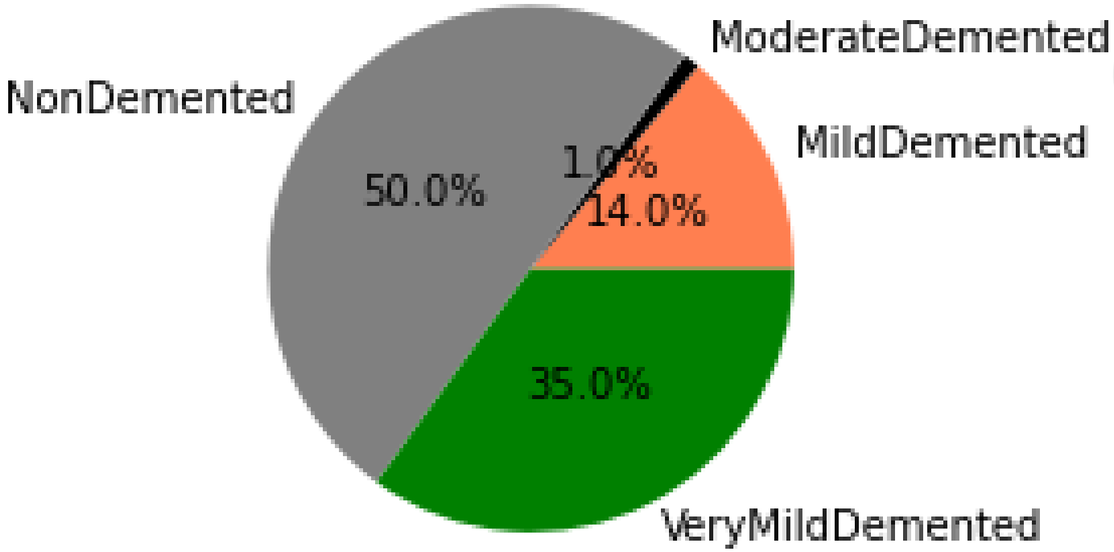
Moderate Demented



Non Demented

Data Distribution





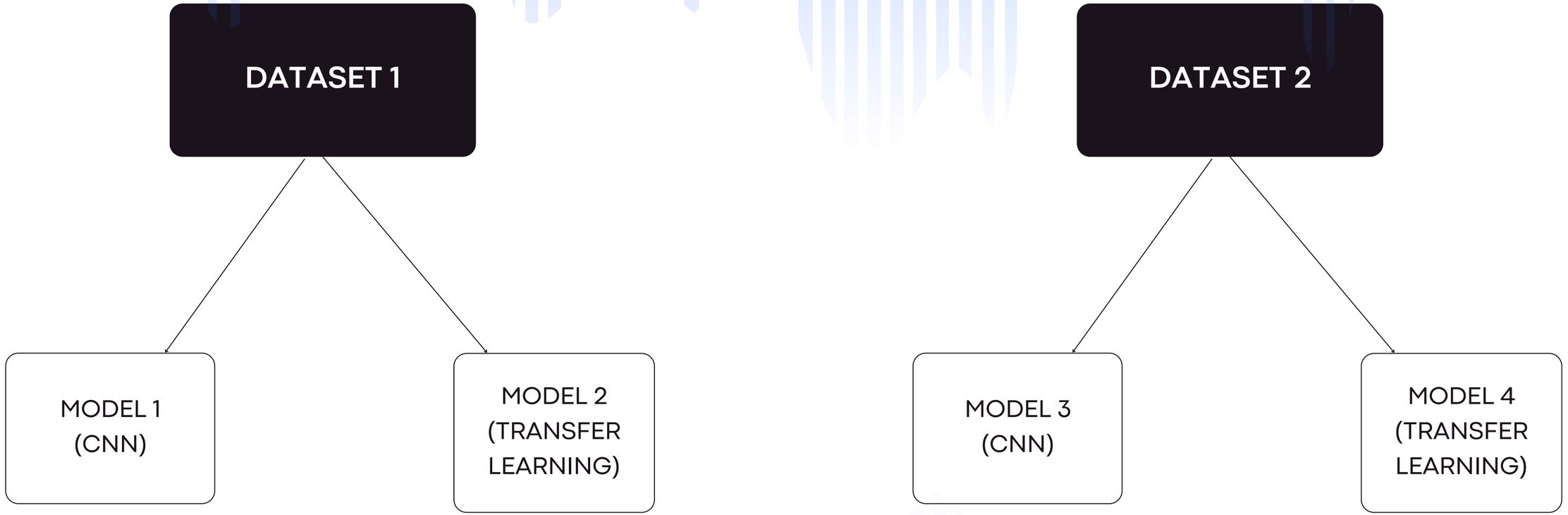
ALGORITHMS USED

CNN

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

TRANSFER LEARNING

Transfer learning (TL) is a technique in machine learning (ML) in which knowledge learned from a task is re-used in order to boost performance on a related task. For example, for image classification, knowledge gained while learning to recognize cars could be applied when trying to recognize trucks.


DATASET 1
DATASET 2
MODEL 1 (CNN)
MODEL 2 (TRANSFER LEARNING)
MODEL 3 (CNN)
MODEL 4 (TRANSFER LEARNING)

Model 1 CNN Dataset 1

Input Layer: tf.keras.Input(shape=IMG_SHAPE)

- This is the input layer of the CNN model. It defines the shape of the input data, which is a grayscale image of dimensions specified by IMG_SHAPE. The image is expected to have a single channel (as it is grayscale).

Hidden Layers:

1. Convolutional Blocks (conv_block function):

- These blocks consist of convolutional, batch normalization, and max-pooling layers. The convolutional layer applies multiple filters to the input image, extracting features.

2. Flatten Layer: tfl.Flatten()

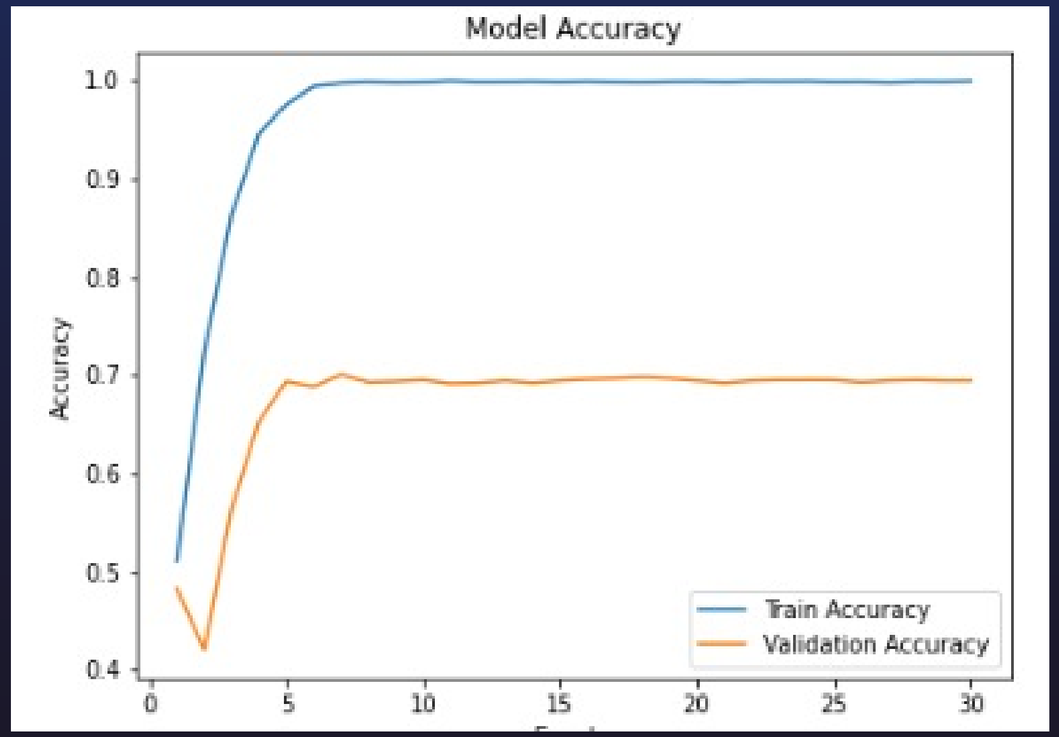
- It transforms the 2D feature maps into a format suitable for passing to the dense (fully connected) layers.

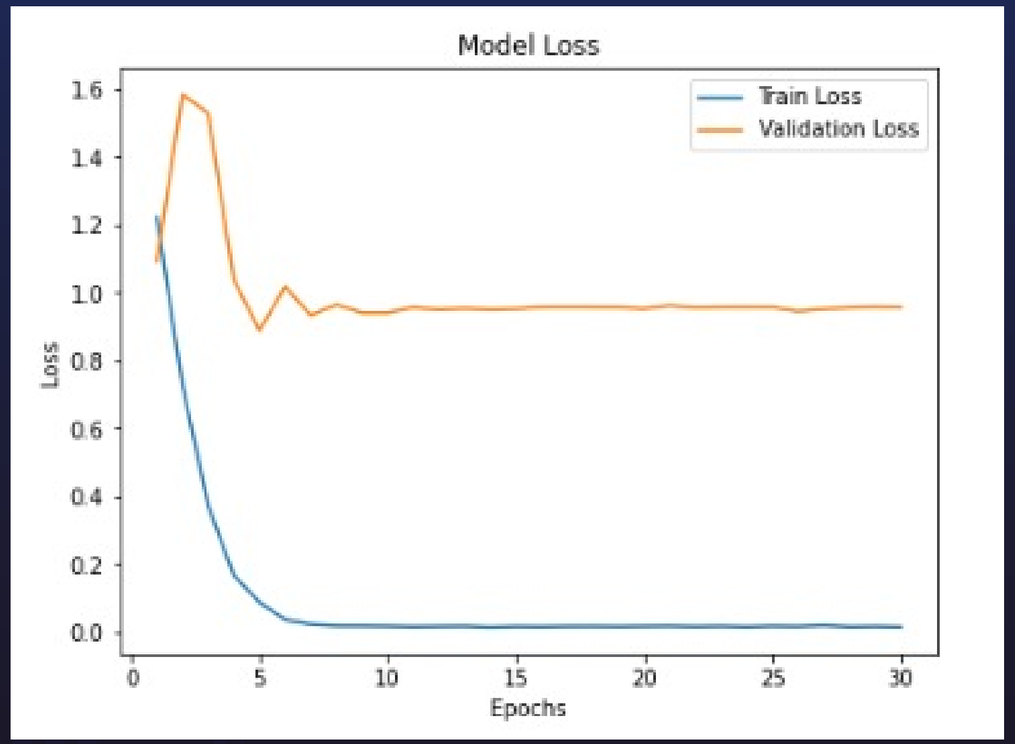
3. Dense Blocks (dense_block function):

- The dense layers process the flattened features obtained from the convolutional layers. Each dense block consists of a dense layer with ReLU activation, followed by batch normalization to stabilize training, and a dropout layer to reduce overfitting

Output Layer: tfl.Dense(units=4, activation='linear', name='output-layer')

This is the output layer of the model. It consists of 4 neurons (units) and uses a linear activation function. The 'linear' activation function means that the output will be directly proportional to the weighted sum of the inputs, without applying any non-linearity. This configuration is suitable for regression tasks, where the model is predicting continuous values. The output layer provides the final predictions of the model.


Model Accuracy
Accuracy
Train Accuracy
Validation Accuracy


Model Loss
Loss
Epochs
Train Loss
Validation Loss

Model 2 Transfer Learning Dataset 1

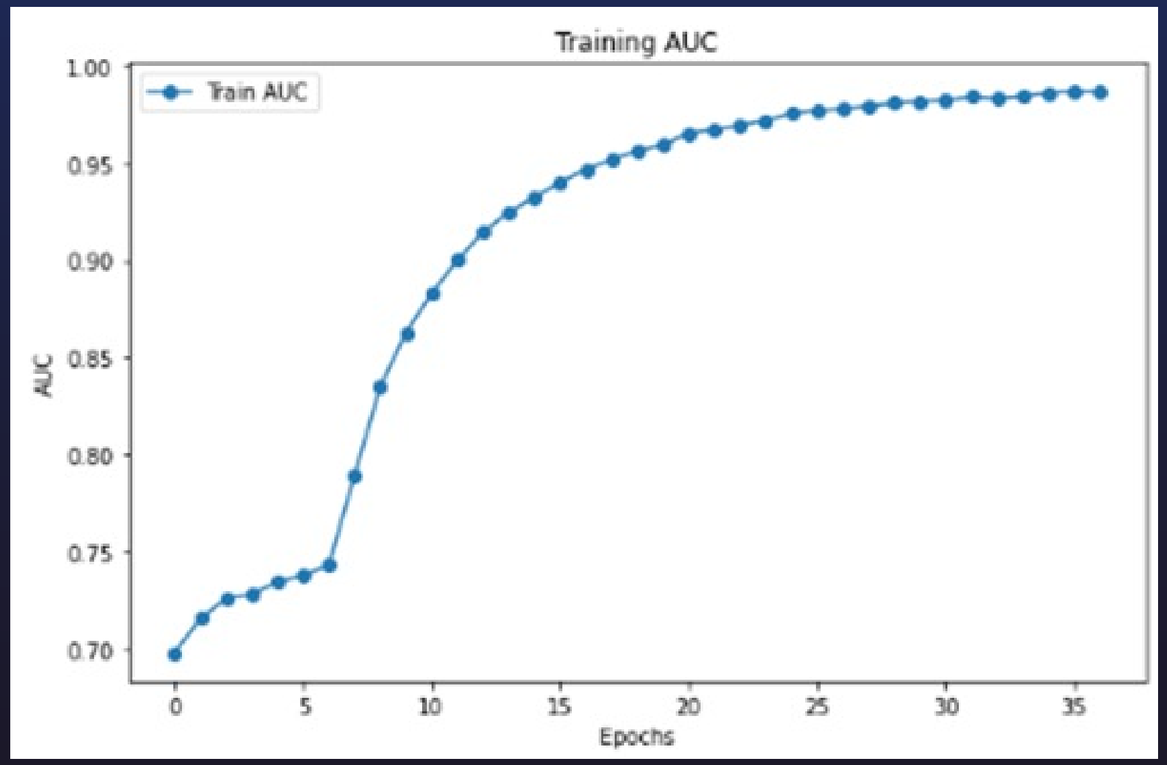
Input Layer: This is the first layer of the model, serving as a placeholder for the input images. It defines the shape of the input data, which is specified as **IMG_SHAPE**.

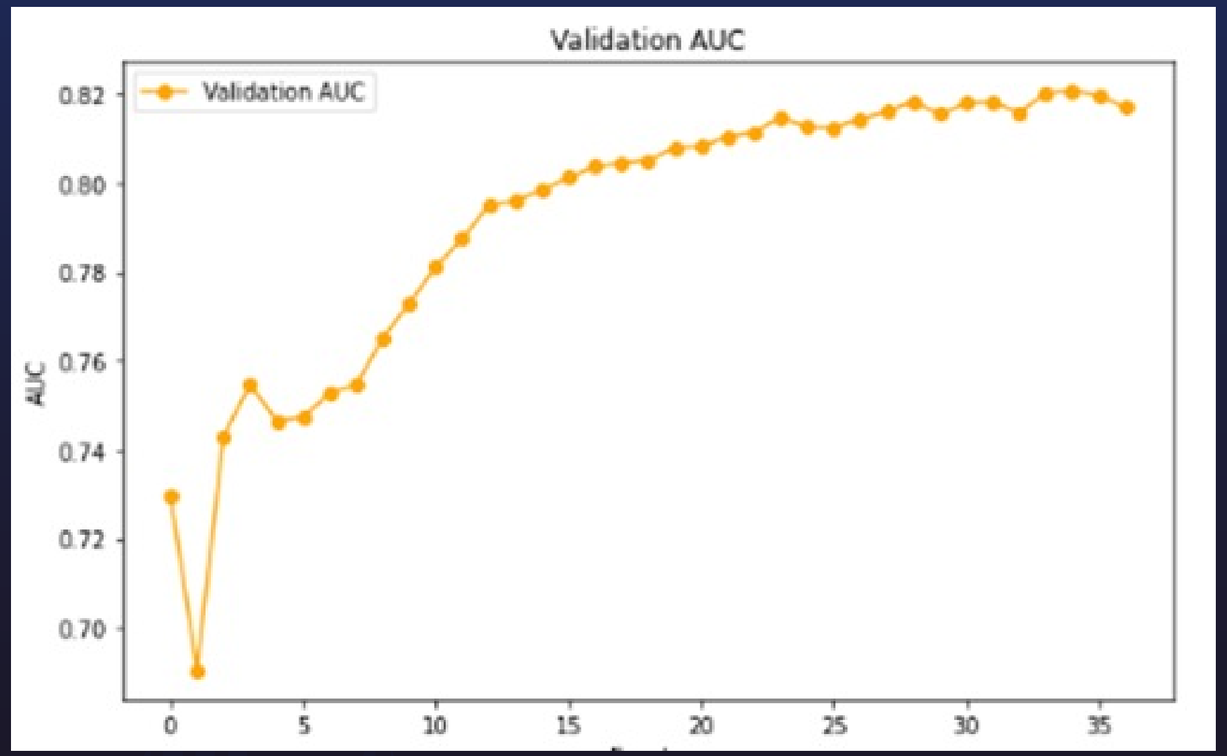
Pre-trained Base Model: The **base_model** is an instance of the InceptionV3 model that comes pre-trained on the ImageNet dataset. It is initialized with the specified **IMG_SHAPE** and **weights='imagenet'**, which loads the pre-trained weights from ImageNet.

Global Average Pooling Layer: After passing the preprocessed input images through the **base_model**, the resulting feature maps are processed by a Global Average Pooling layer. This layer effectively reduces the spatial dimensions to a 1x1 size.

Dropout Layer: To mitigate the risk of overfitting, a Dropout layer is applied after the Global Average Pooling. Dropout randomly deactivates a fraction of the neurons during training.

Dense Layer (Classification Layer):It takes the output from the Dropout layer and connects it to 4 neurons, corresponding to the number of classes in the specific classification task.


Training AUC
Train AUC
AUC
Epochs
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
5
10
15
20
25
30
35


Validation AUC
Validation AUC
AUC
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0
5
10
15
20
25
30
35

Model 3 CNN Dataset 2

Sequential API MODEL:

- **Input Layer**: This layer serves as the entry point for feeding input data into the model.
- **Convolutional Blocks**: These blocks consist of multiple convolutional layers followed by batch normalization and ReLU activation. The model has three convolutional blocks with 16, 32, 64, and 128 filters, respectively. Each convolutional layer processes the feature maps produced by the previous layer, capturing various patterns and features in the data.
- **MaxPooling2D Layer**: It performs max pooling, which reduces the spatial dimensions of the feature maps, retaining the most prominent features. In this model, a MaxPooling2D layer is applied after the first two convolutional layers to downsample the feature maps.
- **Dropout Layers**: These layers help in preventing overfitting by randomly setting a fraction (20% and 20% in this case) of the neurons' outputs to zero during training. This encourages the model to be more robust and generalizable.
- **Flatten Layer**: This layer reshapes the output from the convolutional layers into a flat 1D array, preparing it for the fully connected layers that follow.
- **BatchNormalization layer**: used to normalize the inputs of a layer, making the training process more stable and accelerating convergence.

Fully Connected (Dense) Layers:

- Dense Blocks: These blocks consist of densely connected (fully connected) layers. Each dense block has a specified number of neurons and a dropout rate applied to its output. The dropout rates are set to 0.7, 0.5, and 0.3 for the respective dense blocks.
- Output Layer: The last dense layer has 4 neurons, corresponding to the number of classes in the classification task. The activation function is set to 'softmax', which produces probabilities for each class, indicating the likelihood of the input image belonging to each class.

```
model = Sequential([
        Input(shape=(*IMAGE_SIZE, 3)),
        Conv2D(16, 3, activation=act, padding='same'),
        Conv2D(16, 3, activation=act, padding='same'),
        MaxPool2D(),
        conv_block(32),
        conv_block(64),
        conv_block(128),
        Dropout(0.2),
        conv_block(256),
        Dropout(0.2),
        Flatten(),
        dense_block(512, 0.7),
        dense_block(128, 0.5),
        dense_block(64, 0.3),
        Dense(4, activation='softmax')
    ], name = "cnn_model")
```

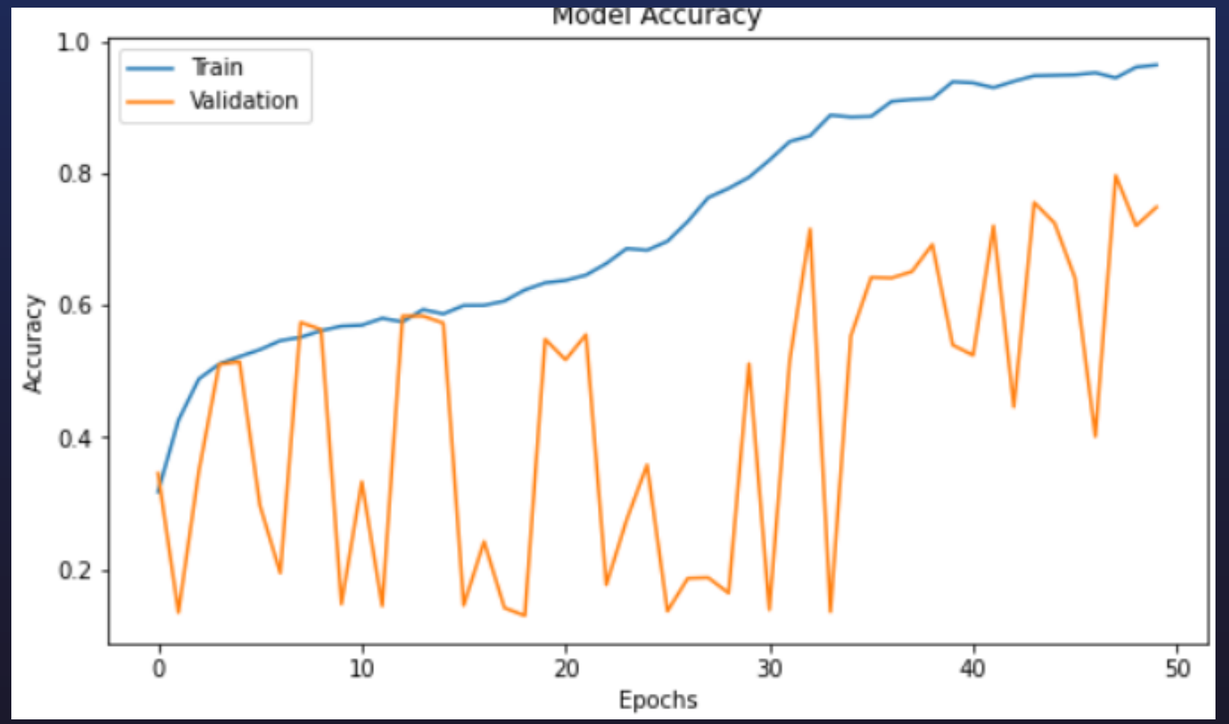
Compilation :

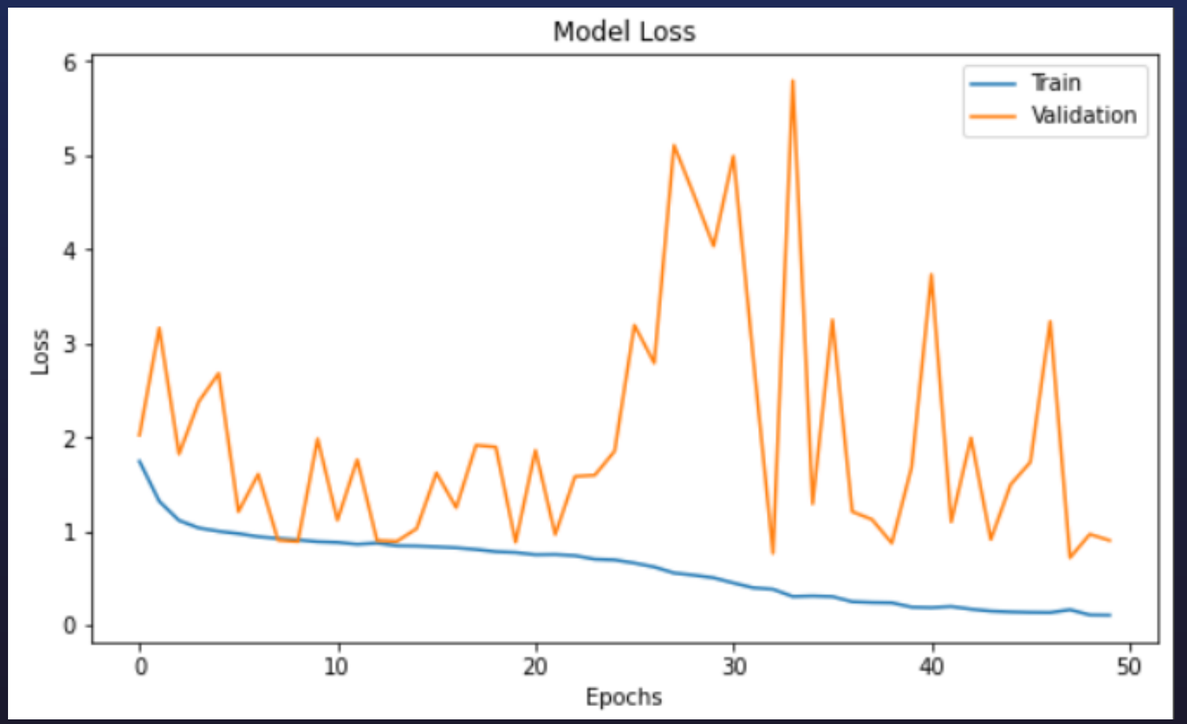
Overall, the code prepares the model for training by compiling it with the specified optimizer, loss function, and evaluation metrics.

1. Metrics: **CategoricalAccuracy** measures overall classification accuracy, **AUC** evaluates area under the ROC curve for binary classification (approximating multi-class performance), and **F1Score** calculates precision-recall balance.
2. Loss Function: **CategoricalCrossentropy** computes the cross-entropy loss between predicted probabilities and one-hot encoded target labels, optimizing the model for multi-class classification tasks.
3. Optimizer: **Adam** optimizes the model's parameters using adaptive learning rates, combining **momentum and RMSprop**. It accelerates convergence and stabilizes training by adjusting learning rates for each parameter adaptively.

Training:

The model is trained using the **fit() method** of the Keras model. The **training data, training labels, validation data, and validation labels** are provided as inputs to the fit() method. Additionally, the number of **epochs** is set to 50, and the **callbacks** list, which includes the EarlyStopping callback, is passed to the fit() method.


Model Accuracy
Train
Validation
Accuracy
Epochs
1.0
0.8
0.6
0.4
0.2
0
10
20
30
40
50


Model Loss
Train
Validation
Loss
Epochs
6
5
4
3
2
1
0
0
10
20
30
40
50

Model 4 Transfer Learning Dataset 2

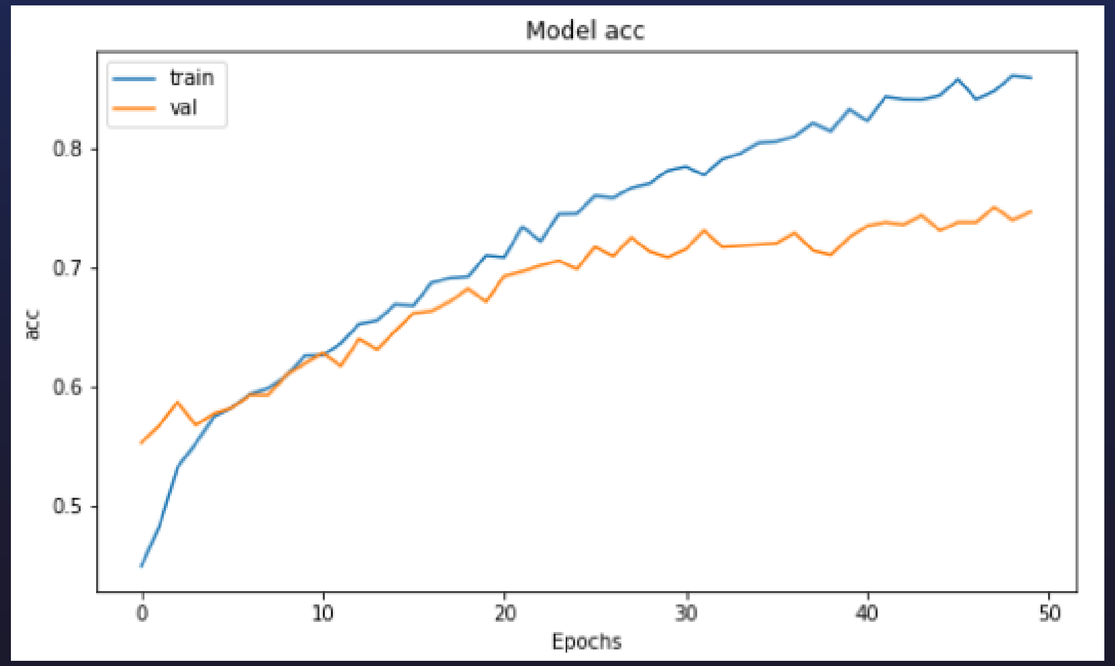
- **InceptionV3 Base Model**: Pre-trained model for feature extraction, enhancing the model's ability to capture meaningful patterns in images.

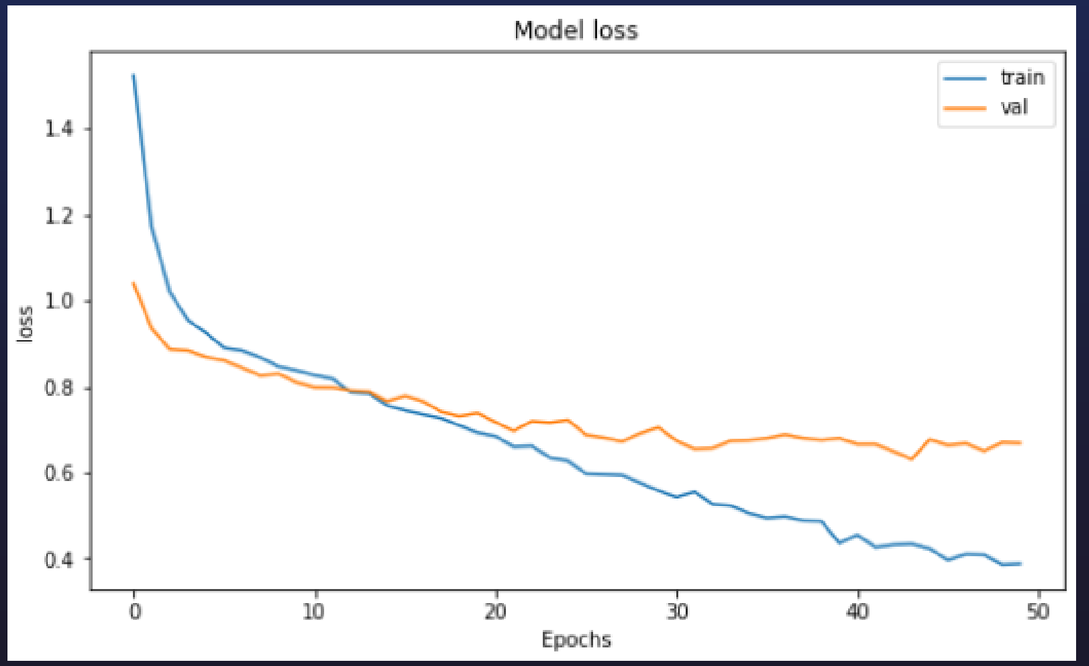
- **Dropout Layer**: Regularization technique that randomly sets 50% of neuron outputs to zero during training to prevent overfitting.
- **Global Average Pooling2D**: Reduces spatial dimensions to 1x1, summarizing feature maps for more efficient computation.

- **Flatten Layer**: Converts the output from the previous layer into a 1D array to prepare for the fully connected layers.
- **BatchNormalization**: Normalizes input to stabilize and accelerate training by reducing internal covariate shift.

- **Dense Layers with Dropout**: Dense layers with ReLU activation (512, 256, 128, 64), each followed by dropout (rate=0.5) for regularization.

- **Output Layer**: Dense layer (4 neurons) with softmax activation for multi-class classification, providing class probabilities.


Model acc
train
val
acc
0.8
0.7
0.6
0.5
0
10
20
30
40
50
Epochs


Model loss
train
val
loss
1.4
1.2
1.0
0.8
0.6
0.4
0
10
20
30
40
50
Epochs

Results & Conclusion

For Dataset 1:

CNN has an accuracy of 73.42%
Transfer Learning Learning has an accuracy of 81.30%

For Dataset 2:

CNN has an accuracy of 80.47%
Transfer Learning has an accuracy of 71.56%

LIMITATIONS

- Lack of interpretability
- Data Bias
- Limited availability of labeled data
- Temporal Dynamics
- Pre Trained Limitations for Transfer Learning
- Model Updating and Validation
- Comorbidity and Multifactoral nature

Future Prospects

- Improved accuracy with larger and diverse datasets.
- Integration of multi-modal data for a comprehensive view of the disease.
- Include temporal dynamics
- Advancements in explainable AI for better model interpretability.
- Personalized medicine approaches based on individual disease variations.
- Integration of predictive models with healthcare systems for early screening.

THANK YOU