République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
ECOLE SUPÉRIEURE EN INFORMATIQUE
8 Mai 1945 - Sidi-Bel-Abbès

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
المدرسة العليا للإعلام الآلي
8 ماي 1945 - سيدي بلعباس

Mini Project Report

# Deep Learning Approaches for Alzheimer's Disease Classification
# - a comparative study -

*Supervised by:*

Dr. Nassima DIF

*Presented by:*

AYAD Amani .....2CS IASD Student
BELMILOUD Maroua .....2CS IASD Student
CHELAOUA Naila .....2CS IASD Student
FETTACHE Dina .....2CS IASD Student
SENKADI Khawla .....2CS IASD Student

GitHub Repository

Submission Date: 09/05/2024

# Contents

# List of Tables



# List of Figures

# 1 Introduction

## 1.1 Background

Alzheimer's disease is a progressive neurodegenerative disorder characterized by cognitive decline, memory loss, and behavioral changes. It is the most common cause of dementia in the elderly population, affecting millions of individuals worldwide. As the global population ages, the prevalence of Alzheimer's disease is expected to increase significantly, posing a growing challenge for healthcare systems worldwide.

## 1.2 Significance of Medical Image Analysis

The early and accurate diagnosis of Alzheimer's disease is crucial for timely intervention and management. However, diagnosing Alzheimer's disease can be challenging, particularly in the early stages when symptoms may be subtle and non-specific. Medical imaging techniques, such as magnetic resonance imaging (MRI) play a critical role in the diagnosis and monitoring of Alzheimer's disease.

## 1.3 Understanding Alzheimer's Severity

Before delving further, it's necessary to explain Alzheimer's in more detail, as the condition encompasses four classes depending on the severity of dementia.

- **Very Mildly Demented:** This stage is characterized by forgetting where one put items, recent acquaintances' names, etc. It's difficult to detect through cognitive ability tests.
- **Mildly Demented:** Patients at this stage struggle to recall words, may get lost en route to destinations, experience loss of focus and work abilities. This is also the stage where patients may forget they are losing memory. Cognitive testing can detect this stage.
- **Moderately Demented:** Symptoms include forgetting recent activities, important historical events, difficulty in budgeting, going outside alone, and loss of empathy. There are further stages within moderate dementia, culminating in the terminal stage where the patient loses mobility and speech.

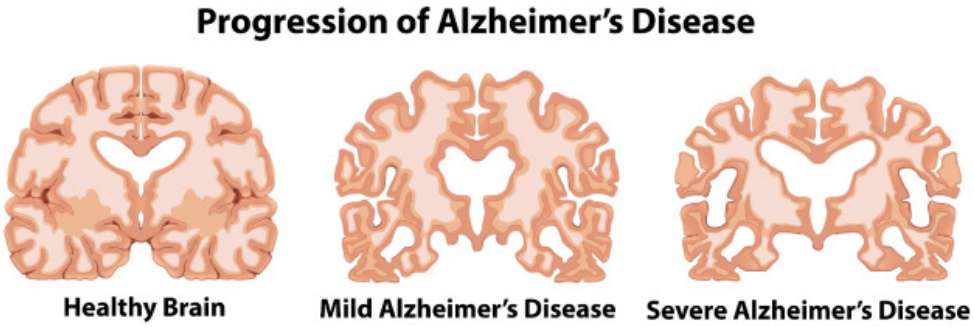


Figure 1: Alzheimer's Progression

## 1.4 Focus of our project

Our study aims to explore a novel approach for medical image classification, specifically targeting Alzheimer's disease diagnosis. Traditionally, deep learning models, including Convolutional Neural Networks (CNNs), have been pivotal in automating and quantifying medical image analysis tasks. In this context, architectures like LeNet and UNet have been widely employed for tasks such as image segmentation, classification, and disease prediction, showcasing remarkable performance. However, the application of CNNs with Long Short-Term Memory (LSTM) networks in medical image classification remains relatively unexplored.

In our project, we investigate the efficacy of LeNet and UNet alongside our proposed CNN-LSTM architecture in Alzheimer's disease classification. By leveraging deep learning methodologies, we aim to enhance the accuracy and efficiency of Alzheimer's disease diagnosis, potentially leading to improved patient outcomes and fostering advancements in Alzheimer's research.

# 2 Dataset Description

## 2.1 Alzheimer MRI Preprocessed Dataset

The data is collected from several websites, hospitals, and public repositories. It consists of preprocessed MRI (Magnetic Resonance Imaging) images, all resized to 128 x 128 pixels. There are four classes of images in the dataset, with a total of 6400 MRI images:

- **Class 1:** Mild Demented (896 images)
- **Class 2:** Moderate Demented (64 images)
- **Class 3:** Non Demented (3200 images)
- **Class 4:** Very Mild Demented (2240 images)

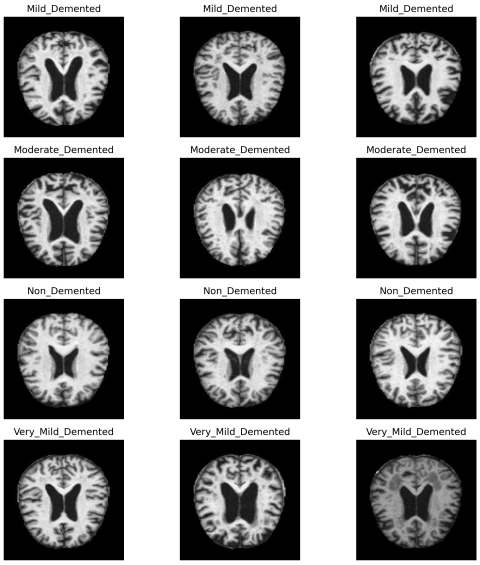


Figure 2: Samples from the Kaggle AD dataset

However, it's noteworthy that the distribution of classes in this dataset is imbalanced. The majority of images belong to the Non_Demented and Very_Mild_Demented classes, while there are relatively fewer images in the Mild_Demented and Moderate_Demented classes. This class imbalance poses challenges during model training and evaluation, potentially leading to bias towards the majority classes and difficulty in accurately predicting the minority classes.

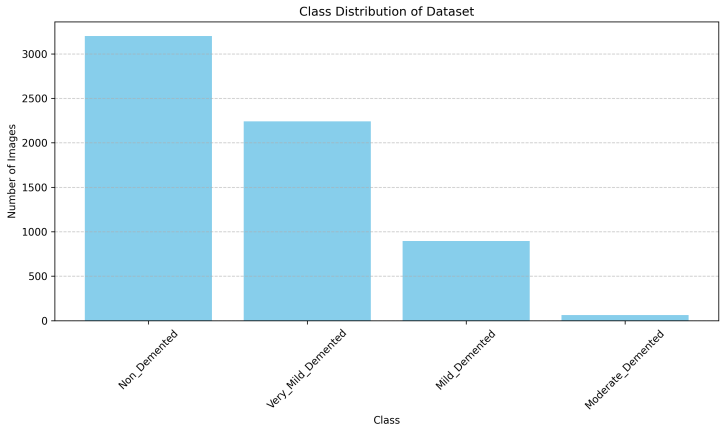


Figure 3: Classes distribution of the Kaggle AD dataset

## 2.2 Data Augmentation

The imbalance in the small dataset leads to overfitting problems during model training, affecting its effectiveness. To address this issue, data augmentation was performed, specifically for the first two classes: Mild Dementia and Moderate Dementia. After augmentation, the total number of images reached 10,672, as shown in Figure 4.

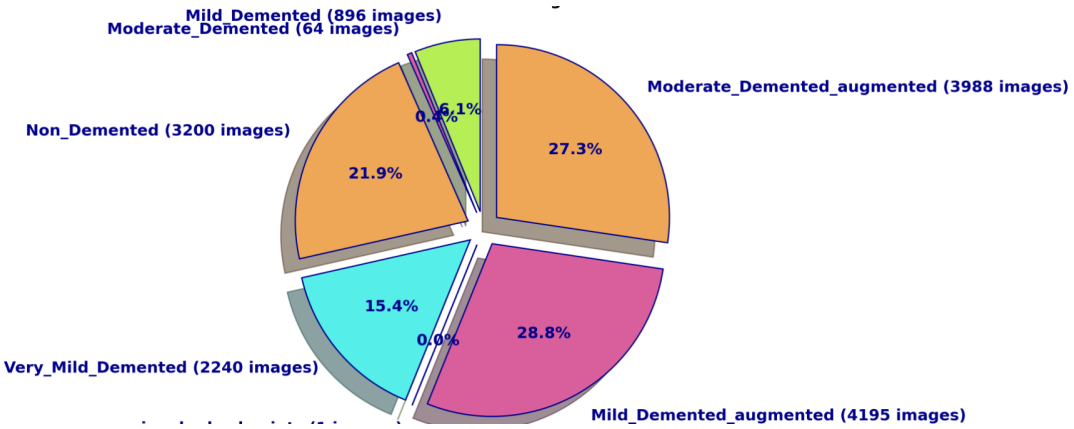


Figure 4: Class distribution of the Kaggle AD dataset after augmentation

## 2.3 Data Augmentation Parameters

In this study, data augmentation was performed using the following parameters:

Table 1: Data Augmentation Parameters

| **Name** | **Value** | **Definition** |
|---|---|---|
| Zoom range | [0.98, 1.02] | This means the images could be zoomed in by up to 2% or zoomed out by up to 2%. |
| Brightness range | [0.8, 1.2] | A range controlling the intensity of the brightness. |
| Horizontal flip | True | A boolean parameter indicating whether horizontal flipping of images is allowed. When enabled, each image has a 50% chance of being horizontally flipped. |
| Fill mode | 'constant' | Specifies the method used for filling in newly created pixels resulting from transformations like rotation . |
| Data format | 'channels_last' | Defines the ordering of dimensions in the input data. This format indicates that the color channels are the last dimension of the input data. |
| Rotation range | 10 degrees | Specifies the range of degrees for random rotations applied to the images. This allows for slight rotations. |
| Shear range | 0.1 | Controls the shearing transformation applied to the images, which distorts the shapes of objects. |
| Width shift range | 0.1 | Specify the range of fractions by which images are horizontally shifted, allowing for translations of the images. This allows for shifts of up to 10% of the image width. |
| Height shift range | 0.1 | Specify the range of fractions by which images are vertically shifted, allowing for translations of the images. This allows for shifts of up to 10% of the image height. |

## 2.4 Data Preprocessing

1. **Images Rescaling**: The pixel values of the images were scaled to the range [0, 1] by dividing each pixel value by 255.0. This rescaling ensures that the pixel values are in a suitable range for training the model.

2. **Data Splitting**: The dataset was split into training, validation, and test sets using a stratified splitting strategy. The training set comprised 70% of the data, while the validation was 24% and test sets comprised 6% of the data. This splitting ensures that the distribution of classes remains consistent across the different sets.

After preprocessing, the dataset was ready for training the deep learning models. The sizes of the training and test sets were as follows:

- **Training set**: 9536 samples
- **Validation set**: 3269 samples
- **Test set**: 817 samples

The class distribution in the dataset after splitting is shown below:

Table 2: Number of samples after splitting into Training, validation & test set

| **Set** | **Class 0** | **Class 1** | **Class 2** | **Class 3** |
|---|---|---|---|---|
| **Train** | 2945 | 2791 | 2242 | 1558 |
| **Validation** | 1013 | 944 | 771 | 541 |
| **Test** | 237 | 253 | 187 | 141 |

Table 3: Number of samples in each class

| **Class** | **Number of Samples** |
|---|---|
| 0 | 4195 |
| 1 | 3988 |
| 2 | 3200 |
| 3 | 2240 |

Table 4: Classes Names according to index

| **Class Name** | **Index** |
|---|---|
| Mild_Demented_augmented | 0 |
| Moderate_Demented_augmented | 1 |
| Non_Demented | 2 |
| Very_Mild_Demented | 3 |

These preprocessing steps ensure that the dataset is appropriately prepared for training the deep learning models, facilitating effective learning and accurate classification.

# 3 Model Architectures

## 3.1 UNet

The UNet algorithm is a convolutional neural network (CNN) architecture commonly used for image segmentation tasks. It is particularly well-suited for medical image analysis, including MRI images, due to its ability to capture fine details and spatial relationships.

### 3.1.1 Why UNet for This Dataset:

UNet was chosen for this dataset due to its effectiveness in segmenting medical images, such as MRI scans. The dataset contains MRI images of brains affected by Alzheimer's disease, and the goal is to accurately classify and segment different regions of interest within these images. UNet's architecture is well-suited for this task as it can handle the complex and detailed structures present in MRI images.

### 3.1.2 How UNet Works:

**Encoder definition**: The encoder part of UNet is similar to the convolutional layers of a CNN, where it extracts features from the input image. However, unlike a typical CNN, UNet's encoder consists of multiple convolutional layers followed by downsampling operations, such as max-pooling. This downsampling reduces the spatial dimensions of the feature maps while increasing the receptive field, allowing the model to capture higher-level features.
**Encoder in our architecture**: The input layer accepts images with the specified input shape, denoted as $X$. Two convolutional layers with 32 and 64 filters, respectively, followed by ReLU activation functions ($\text{ReLU}(x) = \max(0, x)$) and 'same' padding, operate as the encoder. These layers can be represented as:

$$E = \text{Conv}_2(\text{Conv}_1(X))$$

where $\text{Conv}_1$ and $\text{Conv}_2$ represent the convolutional operations.

Two MaxPooling layers, denoted as MaxPool, further downsample the feature maps, yielding encoded features, $E_{\text{encoded}}$:

$$E_{\text{encoded}} = \text{MaxPool}(\text{Conv}_2(\text{Conv}_1(X)))$$

**Decoder definition**: The decoder part of UNet is responsible for upsampling the feature maps back to the original image resolution. It utilizes transposed convolutional layers or upsampling operations to increase the spatial dimensions of the feature maps. Additionally, skip connections from the encoder are incorporated to concatenate feature maps at different scales, aiding in the recovery of fine details during the upsampling process.

**Decoder in our architecture**: The flattened output of the last MaxPooling layer serves as the input for the decoder, denoted as $E_{\text{encoded}}$.

Two dense layers with 128 and 4 units, respectively, followed by ReLU and softmax activation functions, reconstruct the segmented image. These layers transform the extracted features from the encoder into a segmentation map, denoted as $Y$.

The segmentation process can be represented mathematically as:

$$Y = \text{Softmax}(\text{ReLU}(\text{Dense}_2(\text{Dense}_1(E_{\text{encoded}}))))$$

In our architecture, the decoder consists of dense layers instead of transposed convolutional layers. These dense layers perform the upsampling operation and refine the features extracted by the encoder, producing the final segmentation map.

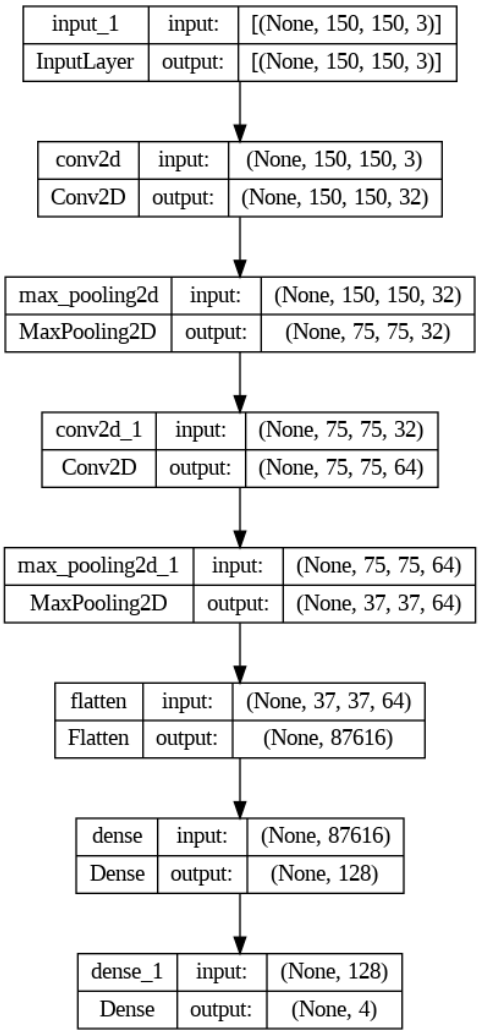


Figure 5: U-Net Model Architecture

## 3.2 LeNet

The LeNet algorithm is a convolutional neural network (CNN) architecture originally designed for handwritten digit recognition. It consists of multiple convolutional and pooling layers followed by fully connected layers, making it suitable for image classification tasks.

### 3.2.1 Why LeNet for This Dataset:

LeNet's design, initially created for handwritten digit recognition, focuses on extracting features from images through convolutional and pooling layers. While its original purpose differs from the dataset's task, LeNet's ability to effectively analyze image features makes it suitable for classifying Alzheimer's disease using MRI images.

### 3.2.2 How LeNet Works:

LeNet follows a straightforward architecture consisting of convolutional, pooling, flattening, and dense layers.

**Convolutional Layers**: LeNet begins with two convolutional layers, each with 6 and 16 filters, respectively. These layers convolve over the input images to extract important features. Let $X$ represent the input image, $C_1$ and $C_2$ denote the outputs of the first and second convolutional layers:

$$C_1 = \text{Conv}_1(X)$$

$$C_2 = \text{Conv}_2(C_1)$$

**ReLU Activation**: After each convolutional layer, the Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity:

$$A_1 = \text{ReLU}(C_1)$$

$$A_2 = \text{ReLU}(C_2)$$

**MaxPooling Layers**: Two MaxPooling layers are utilized to downsample the feature maps, retaining important information while reducing computational complexity:

$$M_1 = \text{MaxPool}(A_1)$$

$$M_2 = \text{MaxPool}(A_2)$$

**Flatten Layer**: The output of the last MaxPooling layer is flattened into a vector to prepare for the fully connected layers:

$$F = \text{Flatten}(M_2)$$

**Dense Layers**: LeNet concludes with three dense layers with 120, 84, and 4 units, respectively. These layers process the flattened features to perform classification. The first two dense layers utilize ReLU activation functions, while the final layer employs a softmax activation function for multi-class classification:

$$D_1 = \text{ReLU}(\text{Dense}_1(F))$$

$$D_2 = \text{ReLU}(\text{Dense}_2(D_1))$$

$$Y = \text{Softmax}(\text{Dense}_3(D_2))$$

Here, $Y$ represents the output classification probabilities for each class.

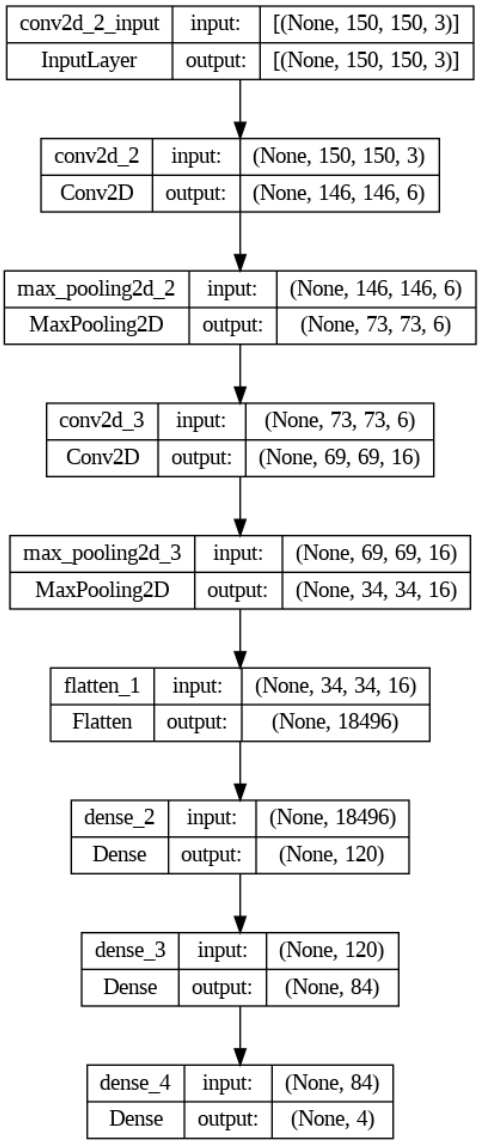


Figure 6: LeNet Model Architecture

## 3.3 LSTM UniDirectionnel

The LSTM Unidirectionnel model is a type of recurrent neural network (RNN) architecture commonly used for sequential data processing tasks. While typically applied to sequences such as text or time series data, it can also be adapted for image data, which inherently possesses spatial properties.

### 3.3.1 Why CNN + LSTM Unidirectionnel for This Dataset:

Although images are inherently spatial data, the model treats the output of the convolutional layers as a sequence of feature vectors. These feature vectors capture spatial information such as edges, textures, and shapes extracted from different regions of the image.

By processing these feature vectors sequentially, the LSTM architecture can capture spatial dependencies and patterns within the image data. This is particularly advantageous for tasks like image classification, where understanding the spatial relationships between features is crucial for accurate classification.

Additionally, the unidirectional nature of the LSTM simplifies the model architecture, focusing solely on learning dependencies in one direction, from past to future. This simplification can potentially improve computational efficiency, making the model faster to train and evaluate.

### 3.3.2 How CNN +LSTM Unidirectionnel Works:

The architecture of the CNN +LSTM Unidirectionnel model consists of convolutional layers followed by LSTM layers, designed to process spatial features extracted from input images.

**Convolutional Layers**: The model begins with a series of convolutional layers, which extract relevant spatial features from the input images. These layers use kernels to perform convolution operations, resulting in feature maps that capture different aspects of the input images.

$$F = \text{Conv}_{32}(X)$$

$$F = \text{Conv}_{64}(F)$$

$$F = \text{Conv}_{128}(F)$$

where $X$ represents the input images, $F$ represents the feature maps, and $\text{Conv}_{32}$, $\text{Conv}_{64}$, and $\text{Conv}_{128}$ denote the convolutional operations with 32, 64, and 128 filters, respectively.

**Flattening and Reshaping**: The output of the convolutional layers is flattened to a one-dimensional vector and reshaped into a 3D tensor, which serves as the input to the LSTM layers. This reshaping step organizes the spatial features extracted by the convolutional layers into a sequence format suitable for LSTM processing.

**LSTM Layers**: The reshaped feature tensor is then fed into two LSTM layers, each followed by a dropout layer for regularization. The first LSTM layer has 256 units and returns sequences, allowing it to capture dependencies within the spatial features. The second LSTM layer has 128 units and does not return sequences, consolidating the learned dependencies for classification.

$$H = \text{LSTM}_{256}(F_{\text{reshaped}})$$

$$H_{\text{final}} = \text{LSTM}_{128}(H)$$

where $H$ represents the output sequences from the LSTM layers, $F_{\text{reshaped}}$ represents the reshaped feature tensor, and $\text{LSTM}_{256}$ and $\text{LSTM}_{128}$ denote the LSTM operations with 256 and 128 units, respectively.

**Output Layer**: Finally, the output sequences from the LSTM layers are passed through a dense layer with softmax activation to produce class probabilities.

The model is trained using categorical cross-entropy loss and optimized using stochastic gradient descent (SGD) with a learning rate of $1 \times 10^{-2}$ and gradient clipping.

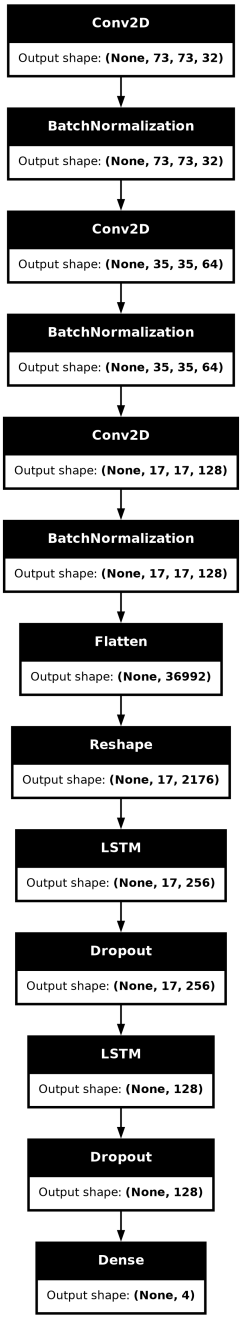


Figure 7: Unidirectional LSTM Architecture

## 3.4 CNN + LSTM BiDirectionnel

### 3.4.1 Why CNN + LSTM Bidirectionnel for This Dataset:

Unlike the unidirectional CNN + LSTM model, which processes data sequentially in one direction, the bidirectional LSTM model examines data in both forward and backward directions. This allows it to capture more context and dependencies in the input data, making it particularly effective for tasks like image analysis, where spatial relationships are important.

### 3.4.2 How CNN + LSTM Bidirectionnel 01 Works:

The Bidirectional LSTM (Bi-LSTM) first model comprises two LSTM layers that process data in both forward and backward directions. It can be applied to image data by treating it as a spatial grid rather than a temporal sequence.

In the architecture provided, convolutional layers are first applied to the input images to extract spatial features. Let $X$ represent the input images, and $F(X)$ denote the feature extraction function performed by the convolutional layers. The output of this process, denoted as $X_{3D}$, is a 3D tensor that encapsulates the spatial features extracted from the images.

$$F(X) \rightarrow X_{3D}$$

The 3D tensor $X_{3D}$ is then flattened and reshaped into a suitable format to serve as input to the Bidirectional LSTM layers. Let $X_{3D}^{\text{flat}}$ represent the flattened tensor, and $X_{\text{reshaped}}$ denote the reshaped tensor. The reshaped tensor $X_{\text{reshaped}}$ has dimensions that allow it to be processed by the Bidirectional LSTM layers.

$$X_{3D}^{\text{flat}} \rightarrow X_{\text{reshaped}}$$

The Bidirectional LSTM layers process the reshaped tensor $X_{\text{reshaped}}$ bidirectionally, capturing both forward and backward dependencies. This allows the model to understand spatial relationships within the input images more comprehensively.

**Forward LSTM:** The forward LSTM layer computes hidden states $\overrightarrow{h}_t$ for each sample $x_t$:

$$\overrightarrow{h}_t = \text{LSTM}(\overrightarrow{x}_t, \overrightarrow{h}_{t-1})$$

**Backward LSTM:** The backward LSTM layer computes hidden states $\overleftarrow{h}_t$ for each sample $x_t$:

$$\overleftarrow{h}_t = \text{LSTM}(\overleftarrow{x}_t, \overleftarrow{h}_{t+1})$$

The outputs of the forward and backward LSTM layers are concatenated to obtain the final output $h_t$:

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$$

By combining convolutional and Bidirectional LSTM layers, the model can effectively learn hierarchical features and capture spatial dependencies within the input images.

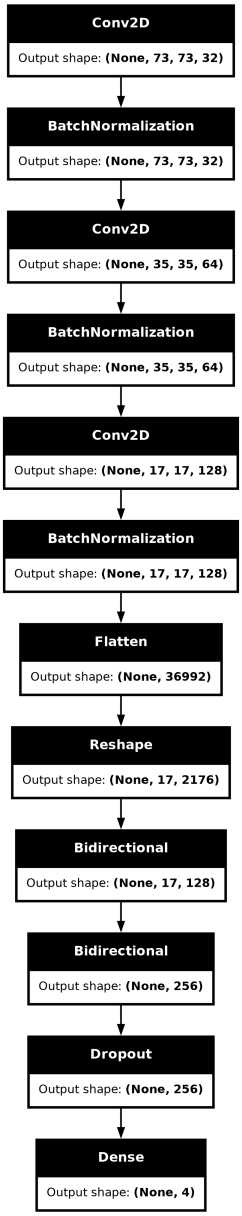


Figure 8: Bidirectional LSTM - First Architecture

### 3.4.3 How CNN + LSTM Bidirectionnel Model 2 Works:

In the architecture provided, convolutional layers are first applied to the input images to extract spatial features. Let $X$ represent the input images, and $F(X)$ denote the feature extraction function performed by the convolutional layers. The output of this process, denoted as $X_{3D}$, is a 3D tensor that encapsulates the spatial features extracted from the images.

$$F(X) \rightarrow X_{3D}$$

The 3D tensor $X_{3D}$ is then flattened and reshaped into a suitable format to serve as input to the Bidirectional LSTM layers. Let $X_{3D}^{\text{flat}}$ represent the flattened tensor, and $X_{\text{reshaped}}$ denote the reshaped tensor. The reshaped tensor $X_{\text{reshaped}}$ has dimensions that allow it to be processed by the Bidirectional LSTM layers.

$$X_{3D}^{\text{flat}} \rightarrow X_{\text{reshaped}}$$

The Bidirectional LSTM layers process the reshaped tensor $X_{\text{reshaped}}$ bidirectionally, capturing both forward and backward dependencies. This allows the model to understand spatial relationships within the input images more comprehensively.

**Forward LSTM:** The forward LSTM layer computes hidden states $\overrightarrow{h}_t$ for each sample $x_t$:

$$\overrightarrow{h}_t = \text{LSTM}(\overrightarrow{x}_t, \overrightarrow{h}_{t-1})$$

**Backward LSTM:** The backward LSTM layer computes hidden states $\overleftarrow{h}_t$ for each sample $x_t$:

$$\overleftarrow{h}_t = \text{LSTM}(\overleftarrow{x}_t, \overleftarrow{h}_{t+1})$$

The outputs of the forward and backward LSTM layers are concatenated to obtain the final output $h_t$:

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$$

By combining convolutional and Bidirectional LSTM layers, the model can effectively learn hierarchical features and capture spatial dependencies within the input images. This approach enhances the model's ability to understand complex spatial relationships, making it well-suited for the classification task on the given dataset.

**Differences from Model 1:**

| **Change** | **Previous Value** | **New Value** | **Reason** |
|---|---|---|---|
| Optimizer | SGD (learning_rate=1e-2, clipnorm=1.) | Adam (learning_rate=0.001) | Changed from SGD to Adam optimizer for potentially better performance. |
| Regularization | None | Kernel Regularizer (L2: 0.01) | Regularization added to LSTM layers in the second model to prevent overfitting. |
| Number of LSTM units in the first LSTM layer | 64 | 128 | Increase the number of units to capture more complex patterns and features. |
| Number of LSTM units in the second LSTM layer | 128 | 64 | Reduce computational complexity and prevent overfitting while maintaining model performance. |

Table 5: Comparison of Changes between Model 1 and Model 2

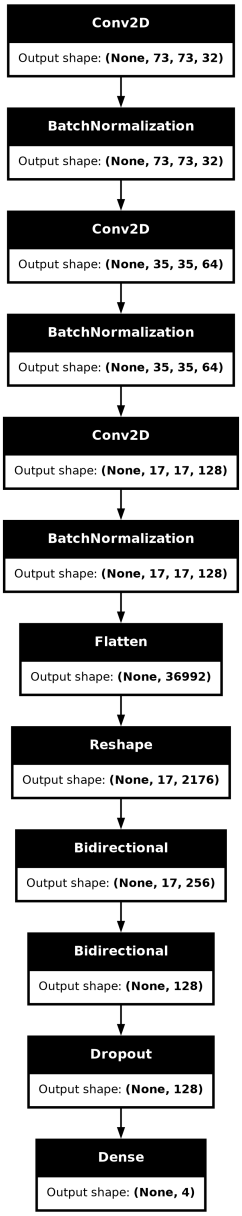


Figure 9: Bidirectional LSTM - Second Architecture

# 4 Model Training & Evaluation

We trained those multiple deep learning models to classify Alzheimer's disease using the prepared dataset. Each model was compiled with the **Adam optimizer** and **categorical cross-entropy loss** function. **Early stopping** was applied based on the loss to prevent overfitting.

## 4.1 LeNet

The training and validation accuracy and loss curves are illustrated in Figure 10.

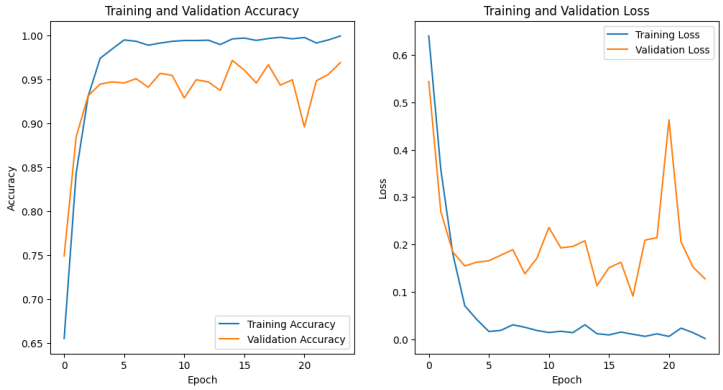


Figure 10: LeNet Training and Validation Curves

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Mild_Demented | 0.95 | 0.92 | 0.93 | 1013 |
| Moderate_demented | 0.92 | 0.94 | 0.93 | 944 |
| Non_Demented | 0.98 | 0.96 | 0.97 | 771 |
| Very_Non_Demented | 0.93 | 0.98 | 0.96 | 541 |
| **Macro Avg** | 0.95 | | | |
| **Weighted Avg** | 0.95 | | | |
| **Validation Accuracy** | 0.95 | | | |
| **Test Accuracy** | 0.967 | | | |
| **Test Loss** | 0.091 | | | |

Table 6: Classification Report for LeNet

### 4.1.1 Discussion

- The LeNet model trained with the Adam optimizer demonstrates a smooth decrease in loss and a corresponding increase in accuracy over the training epochs.
- The use of early stopping, indicated by the sudden stop in training at epoch 24, likely prevented overfitting by halting training when validation loss stopped decreasing.
- The Adam optimizer adapts learning rates for each parameter, which likely contributed to the model's stable convergence. However, there are instances of oscillation in the loss curve, such as at epoch 21, where the loss briefly increases before continuing its downward trend.
- Despite the presence of oscillations, the overall trend of decreasing loss and increasing accuracy suggests effective learning by the model.
- The high test accuracy of 96.70% and low test loss of 0.0907 indicate that the model generalizes well to unseen data, confirming its effectiveness beyond the training and validation sets.

## 4.2 U-Net

The training and validation performance are visualized in Figure 11.

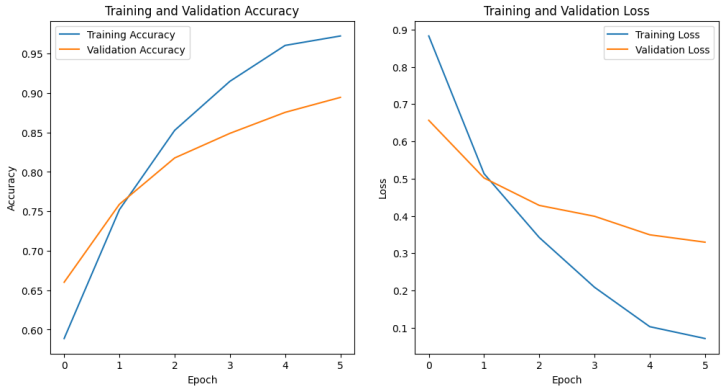


Figure 11: U-Net Training and Validation Curves

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Mild_Demented | 0.76 | 0.35 | 0.48 | 1013 |
| Moderate_demented | 0.56 | 0.88 | 0.68 | 944 |
| Non_Demented | 0.78 | 0.78 | 0.78 | 771 |
| Very_Non_Demented | 0.68 | 0.69 | 0.68 | 541 |
| **Macro Avg** | 0.70 | | | |
| **Weighted Avg** | 0.69 | | | |
| **Validation Accuracy** | 0.66 | | | |
| **Test Accuracy** | 0.6505 | | | |
| **Test Loss** | 0.638 | | | |

Table 7: Classification Report for U-Net

### 4.2.1 Discussion

- The U-Net architecture demonstrates a similar trend in loss reduction and accuracy improvement over the training epochs, as observed with the LeNet model.
- The Adam optimizer, utilized in training the U-Net model, facilitates efficient parameter updates and contributes to the model's convergence.
- The absence of early stopping in the training process might have led to overfitting, indicated by the increasing gap between training and validation performance after epoch 3.
- The decrease in performance on the test dataset, with an accuracy of 65.05% and a loss of 0.6377, compared to the validation set, indicates potential issues with generalization.
- Overfitting could be addressed by implementing early stopping or regularization techniques to prevent the model from fitting noise in the training data excessively.

## 4.3 CNN + UniDirectional LSTM

The training and validation performance are visualized in Figure 12.


Figure 12: CNN + UniDirectional LSTM Training and Validation Curves

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Mild_Demented | 0.95 | 0.93 | 0.94 | 987 |
| Moderate_demented | 0.93 | 0.94 | 0.94 | 942 |
| Non_Demented | 0.97 | 0.89 | 0.93 | 798 |
| Very_Non_Demented | 0.85 | 0.96 | 0.90 | 542 |
| **Macro Avg** | 0.92 | | | |
| **Weighted Avg** | 0.93 | | | |
| **Validation Accuracy** | 0.93 | | | |
| **Test Accuracy** | 0.922 | | | |
| **Test Loss** | 0.223 | | | |

Table 8: Classification Report for CNN + Unidirectional LSTM

### 4.3.1 Discussion

- The CNN with LSTM model demonstrates fluctuations in both loss and accuracy on the validation dataset, indicating potential instability during training.
- Early stopping, a regularization technique, is employed in training the model to prevent overfitting.
- The model is trained using the SGD optimizer with a learning rate of 1e-2 and clipnorm of 1, which helps stabilize the training process by controlling the magnitude of gradients.
- Despite the fluctuations, the model shows a general trend of decreasing loss and increasing accuracy over the training epochs, suggesting that it is learning from the data.
- The performance on the validation set reaches a peak accuracy of approximately 93%, indicating that the model can generalize well to unseen data.
- Performance on the test dataset confirms the model's generalization ability, with an accuracy of approximately 92.18% and a low loss of 0.223.

## 4.4 CNN + BiDirectional LSTM 01

The training and validation performance are visualized in Figure 13.

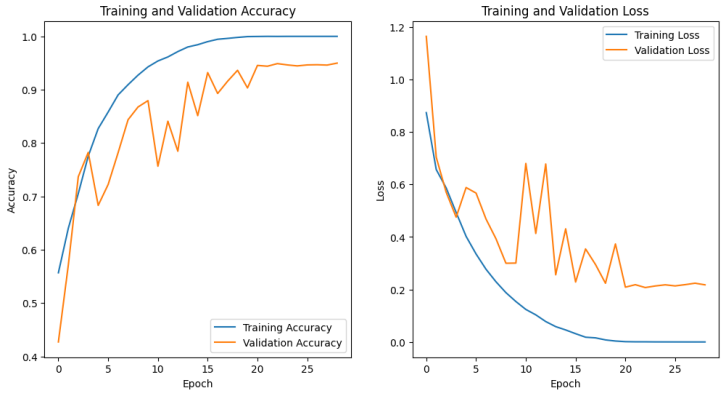


Figure 13: CNN + BiDirectional LSTM 01 Training and Validation Curves

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Mild_Demented | 0.95 | 0.93 | 0.94 | 1007 |
| Moderate_demented | 0.93 | 0.95 | 0.94 | 977 |
| Non_Demented | 0.97 | 0.97 | 0.97 | 751 |
| Very_Non_Demented | 0.95 | 0.95 | 0.95 | 534 |
| **Macro Avg** | 0.95 | | | |
| **Weighted Avg** | 0.95 | | | |
| **Validation Accuracy** | 0.95 | | | |
| **Test Accuracy** | 0.954 | | | |
| **Test Loss** | 0.156 | | | |

Table 9: Classification Report for BiDirectional CNN + LSTM 01

### 4.4.1 Discussion

- The CNN with Bidirectional LSTM model demonstrates fluctuations in both loss and accuracy on the validation dataset, similar to the previous model.
- Early stopping is employed to prevent overfitting, although the model still exhibits fluctuations in performance, indicating the need for further regularization techniques or hyperparameter tuning.
- The model is trained using the SGD optimizer with a learning rate of 1e-2 and clipnorm of 1, similar to the previous model.
- Despite the fluctuations, the model shows a general trend of decreasing loss and increasing accuracy over the training epochs, suggesting that it is learning from the data.
- The performance on the validation set reaches a peak accuracy of approximately 94.98%, indicating that the model can generalize well to unseen data.
- Performance on the test dataset confirms the model's generalization ability, with an accuracy of approximately 95.48% and a low loss of 0.157.

## 4.5 CNN + BiDirectional LSTM 02

The training and validation performance are visualized in Figure 14.

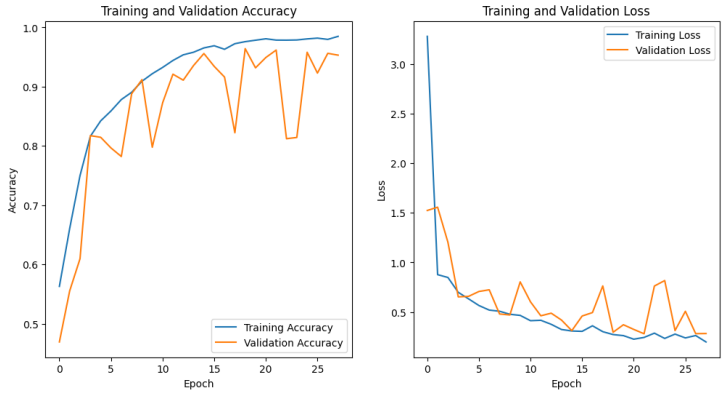


Figure 14: CNN + BiDirectional LSTM 02 Training and Validation Curves

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Mild_Demented | 0.96 | 0.99 | 0.97 | 1024 |
| Moderate_demented | 0.99 | 0.95 | 0.97 | 922 |
| Non_Demented | 0.96 | 0.95 | 0.95 | 770 |
| Very_Non_Demented | 0.92 | 0.95 | 0.94 | 553 |
| **Macro Avg** | 0.96 | | | |
| **Weighted Avg** | 0.96 | | | |
| **Validation Accuracy** | 0.96 | | | |
| **Test Accuracy** | 0.962 | | | |
| **Test Loss** | 0.297 | | | |

Table 10: Classification Report for CNN + BiDirectional LSTM 02

### 4.5.1 Discussion

- The architecture comprises Conv2D layers followed by BatchNormalization, Bidirectional LSTM layers with dropout and kernel regularization to mitigate overfitting and improve generalization.
- Early stopping is utilized to prevent overfitting during training.
- The Adam optimizer with a learning rate of 0.001 is employed for model optimization.
- Despite the introduction of dropout and kernel regularization, fluctuations in both loss and accuracy on the validation dataset are still observed, indicating the complexity of the learning task.
- The model demonstrates a general trend of decreasing loss and increasing accuracy over the training epochs, suggesting that it is learning from the data.
- Performance on the validation set peaks at approximately 96.42% accuracy, demonstrating the model's ability to generalize well to unseen data.
- Evaluation on the test dataset confirms the model's generalization capability, with an accuracy of approximately 96.21% and a relatively low loss of 0.297.

# 5 Conclusion

In summary, we have trained and evaluated five different models for image classification tasks. Here's a comparative overview of their performance:

| Model | Test Accuracy | Test Loss |
|---|---|---|
| LeNet | 0.9670 | 0.0908 |
| UNET | 0.6505 | 0.6377 |

Table 11: CNNs Performance on Test Set

| Model | Test Accuracy | Test Loss |
|---|---|---|
| CNN + Bi_Lstm_02 | 0.9621 | 0.2973 |
| CNN + Bi_Lstm_01 | 0.9548 | 0.1565 |
| CNN + UNI_LSTM | 0.9218 | 0.2230 |

Table 12: Comparison of CNN-LSTM Models Performance on Test Set

Overall, it's evident that deeper architectures and the incorporation of bidirectional LSTM layers have led to substantial improvements in model performance, with Model 5 demonstrating the highest accuracy on the test set. However, further fine-tuning and exploration of regularization techniques could potentially enhance the performance of all models.

After careful evaluation, we selected Model 5 (CNN with Bidirectional LSTM 02) for deployment due to its superior performance on the test set. This model was deployed in a user-friendly interface, allowing users to upload images for classification. Below are some screenshots showcasing the interface and its functionality:

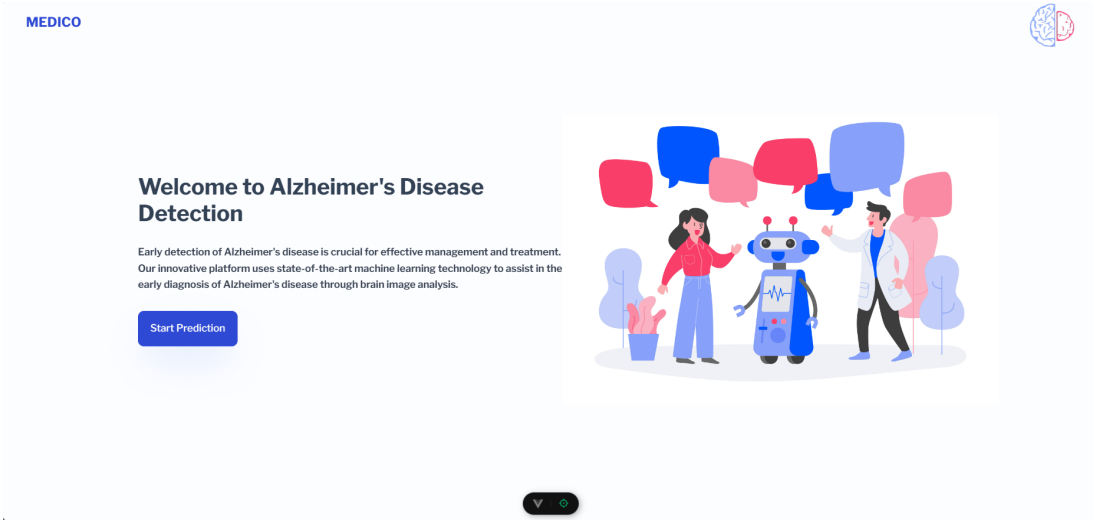


Figure 15: Screenshot 1: Interface Homepage

Code repository :
GitHub Repository

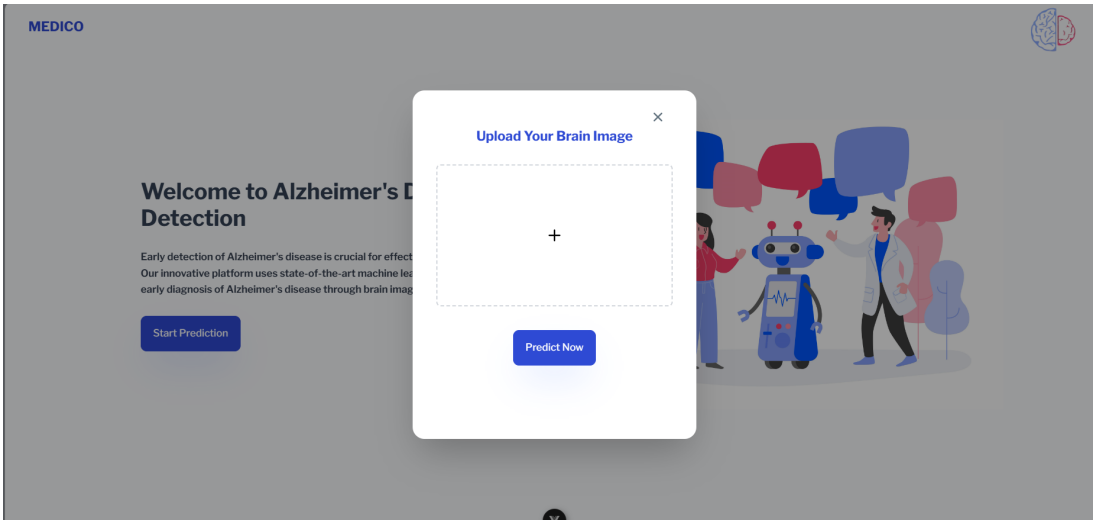


Figure 16: Screenshot 2: Image Upload Page

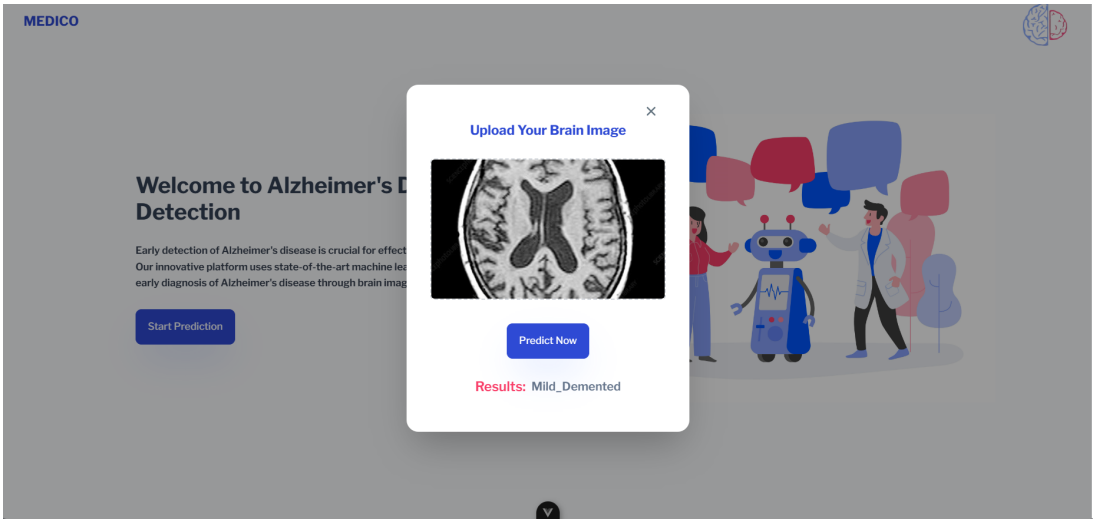


Figure 17: Screenshot 3: Image Prediction Page

# Refrences

# References


[1] Venugopalan, J., Tong, L., Hassanzadeh, H.R., et al. Multimodal deep learning models for early detection of Alzheimer's disease stage. *Sci Rep* 11, 3254 (2021). https://doi.org/10.1038/s41598-020-74399-w

[2] P. Ni, J. Sheng, L. Jiang, and G. Xu. "Sequential ISAR Images Classification Using CNN-Bi-LSTM Method," *2022 3rd China International SAR Symposium (CISS)*, Shanghai, China, 2022, pp. 1-5. https://doi.org/10.1109/CISS57580.2022.9971386

[3] Tatsunami, Y., Taki, M. "Sequencer: Deep LSTM for Image Classification." Rikkyo University, Tokyo, Japan.

[4] Sahoo, A.R., Chakraborty, P. "Hybrid CNN Bi-LSTM neural network for Hyperspectral image classification." Department CSIT, SOA University, Bhubaneswar, Odisha, India.

[5] Benyahia, A., Bahi Azzououm, A., Benammar, A., Mostefaoui, F.Z., Araar, I.E., Taleb, N. "Convolutional Neural Network Deep Learning Approach for Alzheimer Disease Classification." Research Center In Industrial Technologies - CRTI, Algiers, Algeria.

[6] Benyahia, A., Benammar, A., Araar, I.E. "Deep Transfer Learning to Predict Alzheimer Disease Status." Research Center In Industrial Technologies - CRTI, Algiers, Algeria.