Department of Maths and Computer Science
University of Catania

Machine Learning
Exam Project

Binary Classification of MRI Scans for the Diagnosis of Alzheimer's Disease

---

Author:
Antonio Scardace

Submitted to:
Prof. G.M. Farinella

July 2024

Contents

1 Introduction **3**
1.1 Exam Project Description . . . 3

2 Dataset **5**
2.1 Images Preprocessing . . . 6
  2.1.1 Bias-Field Correction . . . 6
  2.1.2 Affine Registration . . . 7
  2.1.3 Skull Stripping . . . 7
  2.1.4 Intensity Normalization . . . 7
2.2 Dataset Split . . . 8
2.3 Dataset Study . . . 8
2.4 Folder Structure of the Dataset . . . 13

3 Proposed Solution **14**
3.1 Model . . . 14
3.2 Loss Function . . . 14
3.3 Optimizer for Learning . . . 15

4 Model Evaluation **16**
4.1 Accuracy Score . . . 16
4.2 Confusion Matrix . . . 16
4.3 AUC of the ROC Curve . . . 17
4.4 Precision . . . 17
4.5 Recall . . . 17
4.6 F1-Score . . . 18
4.7 Comparison with the State-of-the-Art . . . 18

5 Experiments **19**
5.1 Experiment 1 . . . 19
5.2 Experiment 2 . . . 21

6 Source Code **23**
6.1 Jupyter Notebooks . . . 24
6.2 Python Source Code . . . 24

7 Demo **26**

8 Conclusions **27**

1 Introduction

Machine Learning, a subset of artificial intelligence, involves the development of algorithms that enable computers to learn from and make decisions based on data. Among the various applications of ML, image classification stands out as a critical area, where Convolutional Neural Networks (CNNs) have proven exceptionally effective. CNNs, a class of deep learning models, are specifically designed to process and analyze visual data, making them ideal for tasks involving image recognition and classification.

In the context of **Medical Imaging**, ML and CNNs have the potential to revolutionize the diagnosis and treatment of diseases. Medical Imaging encompasses a wide range of techniques and technologies used to create visual representations of the body's interior for clinical analysis and medical intervention. The ability of ML models to analyze these images with high accuracy can significantly enhance the early detection and diagnosis of various medical conditions, leading to better patient outcomes. By automating the interpretation of complex medical images, ML algorithms can assist clinicians in making more accurate diagnoses, reducing the likelihood of human error, and allowing for personalized treatment plans. The ongoing advancements in this field hold promise for improved diagnostic tools, more efficient workflows in medical facilities, and ultimately, a higher standard of care for patients worldwide.

I have thoroughly enjoyed immersing myself in the field of Medical Imaging, particularly in the specific task of my project. This area of study is deeply personal to me due to family reasons, and I am passionate about its potential. I firmly believe that Medical Imaging is one of those fields with a true social impact, ready to benefit both the poor and the rich without discrimination.

1.1 Exam Project Description

This project aims to develop a classifier for Brain MRI scans to differentiate between **Cognitively Normal** individuals **(CN)** and those with **Alzheimer's Disease (AD)**. This entails designing a model to accurately classify and distinguish 3D images corresponding to these two categories. I proposed this image classification task to Professor Giovanni Maria Farinella as part of the Machine Learning course examination. Throughout the development and implementation of the project, I received supervision and guidance from PhD student Lemuel Puglisi.

Classifying Brain MRI scans to distinguish between CN individuals and those with AD presents a significant challenge. This difficulty arises from the variability in image acquisition, which can vary by the capture year or the scanner type used. Developing a robust classification system that can generalize effectively across these differing conditions is crucial. Such a system must be resilient to variations in imaging protocols to ensure accurate and reliable diagnosis.

The project encompasses various stages of the machine learning pipeline, from data obtaining and preprocessing to model training, evaluation, and inference. The repository which serves as the central hub for all project-related files, including code, documentation, and results, is available at `https://github.com/antonioscardace/Tina`.

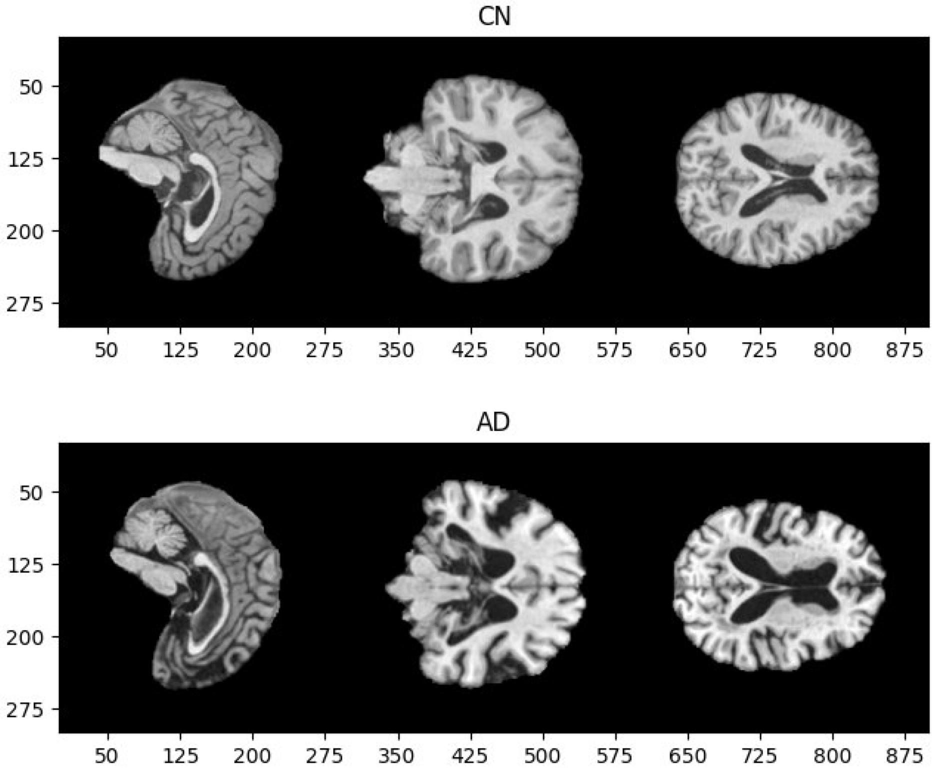


Figure 1: Example of two patients in a different condition. The first one is a CN patient, the second one is an AD patient. In the AD patient, the MRI scan reveals atrophy in brain regions associated with memory and cognition, increased ventricular size and a general loss of brain mass.

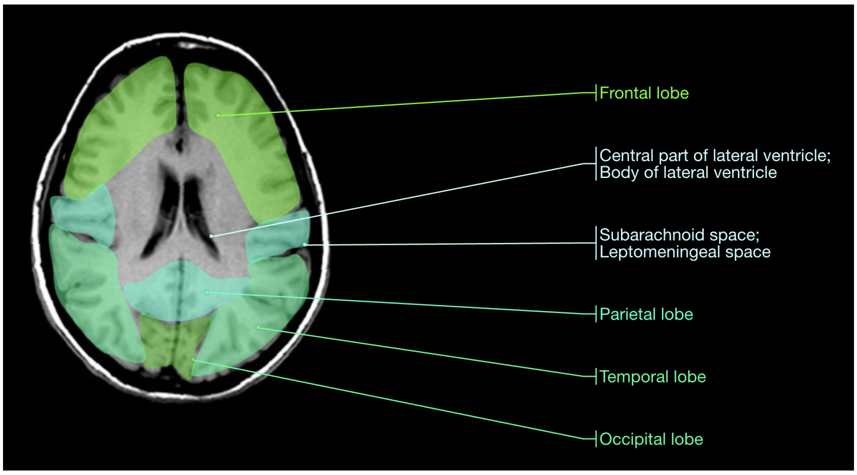


Figure 2: In this axial MRI image, several key brain structures are visible. We can see the **frontal lobe**, which is responsible for decision-making and controlling behaviour and emotions; the **parietal lobe**, which integrates sensory information and is essential for spatial orientation and navigation; the **temporal lobe**, crucial for processing auditory information and involved in memory and speech; and the **occipital lobe**, which handles visual processing, allowing us to interpret visual stimuli. In the centre, the **central ventricles** are visible, and surrounding these is the **subarachnoid space**, filled with cerebrospinal fluid that acts as a protective buffer.

2 Dataset

The dataset utilized in this project was not personally created but rather sourced from an existing collection. This decision was made because generating such data manually is practically impossible for me. The process requires access to MRI machines and a cohort of patients to undergo these scans, both of which are beyond my reach.

Among the various datasets available for tasks involving brain MRI scans, such as the **OASIS 3** dataset, I opted for the **Image and Data Archive (IDA)** managed by the **University of Southern California**. Specifically, I applied for access to the **Alzheimer's Disease Neuroimaging Initiative (ADNI)** study through their selection form at `https://ida.loni.usc.edu/login.jsp`. The ADNI dataset is renowned for its comprehensive collection of neuroimaging and clinical data aimed at investigating Alzheimer's Disease (AD) and other cognitive disorders. You can find it at `https://adni.loni.usc.edu/`.

Upon obtaining access, it was imperative to meticulously select the appropriate data for download. The IDA dataset is extensive, encompassing various acquisition types such as MRI, CT scans, and a multitude of pathological conditions beyond just Cognitively Normal (CN) and Alzheimer's Disease (AD). Therefore, I concentrated on selecting **T1-weighted** MRI scans from the ADNI study, specifically targeting the AD and CN classes.

This careful selection process was crucial to ensure that the dataset was tailored to the specific requirements of this project, enabling a focused analysis of the differentiation between Cognitively Normal individuals and those diagnosed with Alzheimer's Disease. By leveraging the IDA's robust and well-documented dataset, the project benefits from high-quality, standardized data that significantly enhances the reliability and validity of the resulting model.

Once the collection of samples was created on the IDA website, I proceeded to download the **CSV** files and undertook a thorough data cleaning process. During this process, I renamed the columns to ensure consistency and clarity and filtered the scans to include only those performed with the MP-RAGE technique, a widely used protocol for high-resolution brain imaging. A significant characteristic of the ADNI dataset is that each patient typically has multiple scans over several years, often with one scan per year. This abundance of data, while comprehensive, posed challenges. The dataset became excessively large and included many redundant scans, as changes from year to year were often minimal. To address this, I decided to retain, for each patient, at most two scans: the earliest available scan and the most recent one. This approach reduced redundancy while preserving essential longitudinal information. Consequently, my curated dataset was decreased from 29623 samples to 2074 **unique samples**.

ADNI
Alzheimer's Disease
Neuroimaging Initiative

2.1 Images Preprocessing

Once the collection of relevant MRI scans, refined through the previous sampling and cleaning phase, was created on the IDA website, I proceeded to download them. These scans are provided as numerous **DCM** files within a complex folder structure, all zipped for convenience. Consequently, the first step was to automate the unzipping and preprocessing process through bash scripts. The downloaded **ZIP** files were placed in the project path `/data/images/`, and various scripts were executed sequentially:

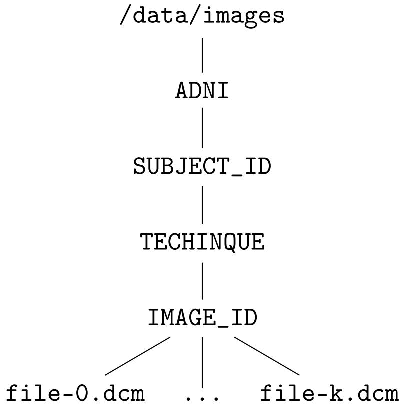
1. `00-extract.sh`: This script is responsible for extracting the downloaded ZIP files into a primary directory named ADNI. That ensures that all files are available for further processing.

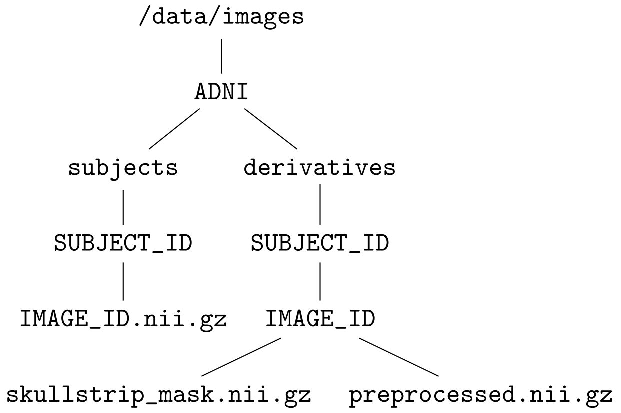
2. `01-organise.sh`: Following extraction, this script simplifies the folder structure, organizing the files in a manner that facilitates subsequent script operations. This reorganization is crucial for maintaining a manageable file hierarchy.

3. `02-transform.sh`: This script transforms the extracted DCM files into a single compressed **NIfTI** file (nii.gz). After the conversion, this script further simplifies the folder structure.

4. `03-preproc.sh`: The final preprocessing script executes 4 preprocessing tools on the scans. As each preprocessing step is completed, intermediate scans are deleted to conserve space, retaining only the output from the last preprocessing tool. This ensures efficient storage management and prepares the data for analysis.

Here is the structure of the folders in `/data/images/` after the initial unzipping step (on the left), and after the final preprocessing step (on the right):





2.1.1 Bias-Field Correction

Bias-Field signal is a low-frequency and very smooth signal that corrupts MRI images especially those produced by old MRI scanners. Image processing algorithms such as segmentation, texture analysis or classification that use the grey-level values of image pixels will not produce satisfactory results. A pre-processing step is needed to correct for the bias field signal before submitting corrupted MRI images to such algorithms or the algorithms should be modified. [1]

Bias-field correction is performed using the **N4** algorithm from the **ANTs (Advanced Normalization Tools)** toolkit. The N4 algorithm corrects these biases by estimating and removing the slowly varying intensity non-uniformities, thereby enhancing the consistency and accuracy of the intensity values across the image.

2.1.2 Affine Registration

Affine registration is a crucial preprocessing step in neuroimaging that addresses the problem of aligning MRI scans to a common coordinate system. Due to variations in patient positioning during scanning, different scans may be misaligned, making direct comparison challenging. Affine registration uses linear transformations to map the brain images into a standardized space. By applying these transformations, this step ensures that anatomical structures are consistently located across all images, facilitating accurate analysis and comparison.

Affine registration is conducted using the **antsRegistrationSyNQuick** script from the **ANTs** toolkit, specifically employing the **SyN (Symmetric Normalization)** algorithm. The target image I used is the **MNI152 T1 template**. You can find and download it at `https://github.com/Jfortin1/MNITemplate/tree/master/inst/extdata`.

2.1.3 Skull Stripping

Skull stripping, also known as brain extraction, is essential for isolating the brain from non-brain tissues such as the skull, scalp, and other surrounding structures in MRI scans. The presence of these non-brain tissues can introduce noise and confound the analysis. Skull stripping employs algorithms that segment and remove these extraneous parts, leaving only the brain tissue. This preprocessing step enhances the focus on relevant brain structures.
For privacy, skull stripping is also required to anonymize a patient as they may be recognized by the shape of their skull or other details external to the brain.

Skull stripping, or brain extraction, is conducted using the **HD-BET (Highly Efficient Brain Extraction Tool)** tool [2]. HD-BET is a product on `https://github.com/MIC-DKFZ/HD-BET` of a collaborative effort between the Department of Neuroradiology at Heidelberg University Hospital and the Division of Medical Image Computing at the German Cancer Research Center (DKFZ).

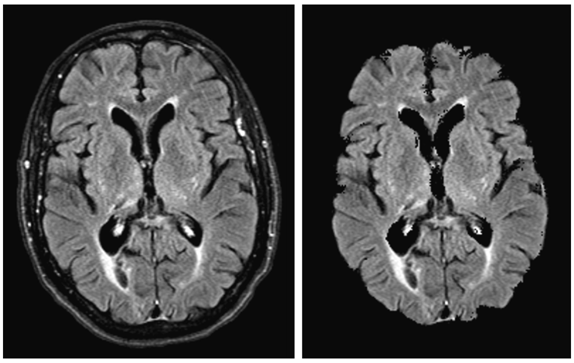


Figure 3: Example of Skull Stripping Application.

2.1.4 Intensity Normalization

Intensity normalization addresses the variability in signal intensity within and between MRI scans. Factors such as scanner differences, acquisition parameters, and patient-specific characteristics can lead to inconsistent image intensity ranges, complicating the comparison and analysis. Intensity normalization standardizes the intensity values within each scan to a common scale, typically by adjusting the mean and standard deviation. This step ensures that the intensity values are comparable across all images, enhancing the reliability of the following analytical processes.
Intensity normalization is performed using the **WhiteStripe** tool from the **intensity-normalization** package. You can find it on `https://github.com/jcreinhold/intensity-normalization`.

2.2 Dataset Split

The dataset was divided into three distinct subsets: **training (TR)**, **validation (VA)**, and **test (TE) sets**. Specifically, **60%** of the dataset was allocated to the training set, **20%** to the validation set, and the remaining **20%** to the test set. This split was conducted randomly so the subsets maintain the original dataset's diversity and characteristics, thereby improving the reliability and robustness of the model evaluation process. This **random allocation** minimizes potential biases and ensures that the model is tested against a comprehensive and varied sample of data, ultimately leading to a more accurate and generalizable machine learning model.

The training set, comprising 60% of the data, is utilized to train the machine learning model. The validation set, consisting of 20% of the data, plays a crucial role in evaluating the model during the training phase. It helps in tuning hyperparameters and preventing overfitting by providing an unbiased evaluation of the model's performance. Finally, the test set, which also comprises 20% of the data, is used to assess the final performance of the model once the training is complete. This provides an objective measure of the model's generalization capability to unseen data.

2.3 Dataset Study

Here are several charts that analyze the **distribution** of the data within the overall dataset, as well as within the training, validation, and test sets. They illustrate various **correlations** and the representativeness and **balance** of the data across the different subsets. These analyses are useful to ensure that the model trained on this data will generalize well to unseen data. Understanding the distribution of the data helps in identifying any potential biases.

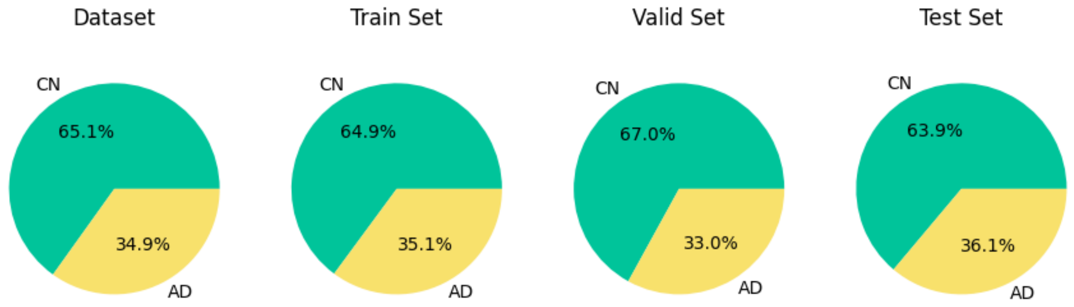


Figure 4: The distribution of diagnoses in the dataset. All sets are balanced. As we can easily see, the number of cognitively normal patients is greater than AD patients.

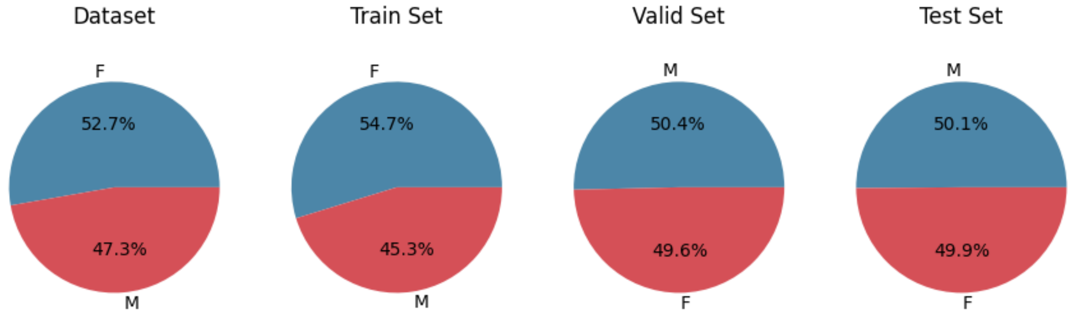
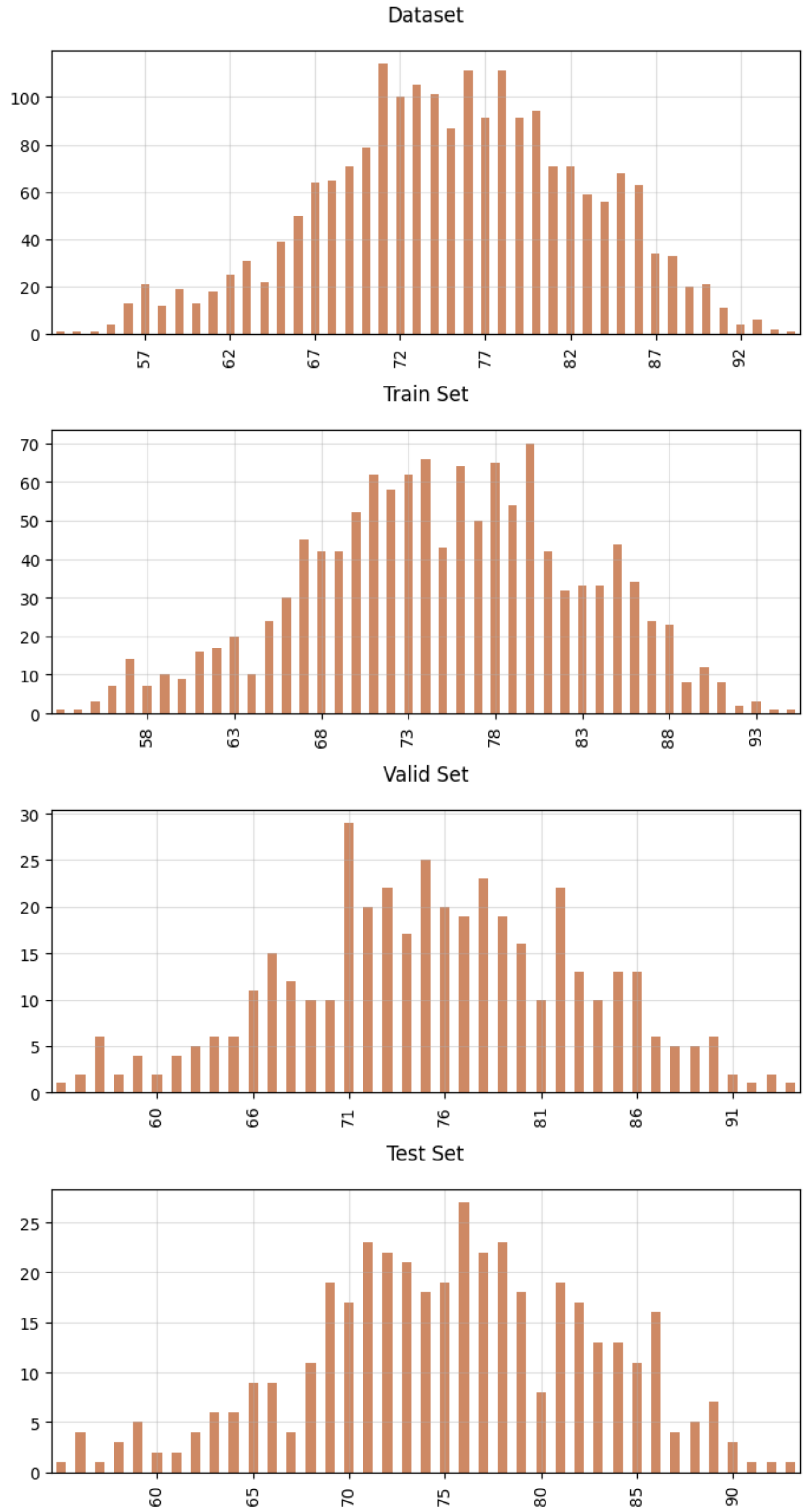


Figure 5: The gender distribution in the dataset. All the sets are balanced.


Dataset

Train Set

Valid Set

Test Set

Figure 6: The age distribution in the dataset. All sets are almost balanced. The **x-axis** denotes age, while the **y-axis** represents the count. The **range** with the most patients is between almost 70 and a little bit more than 80 years old.

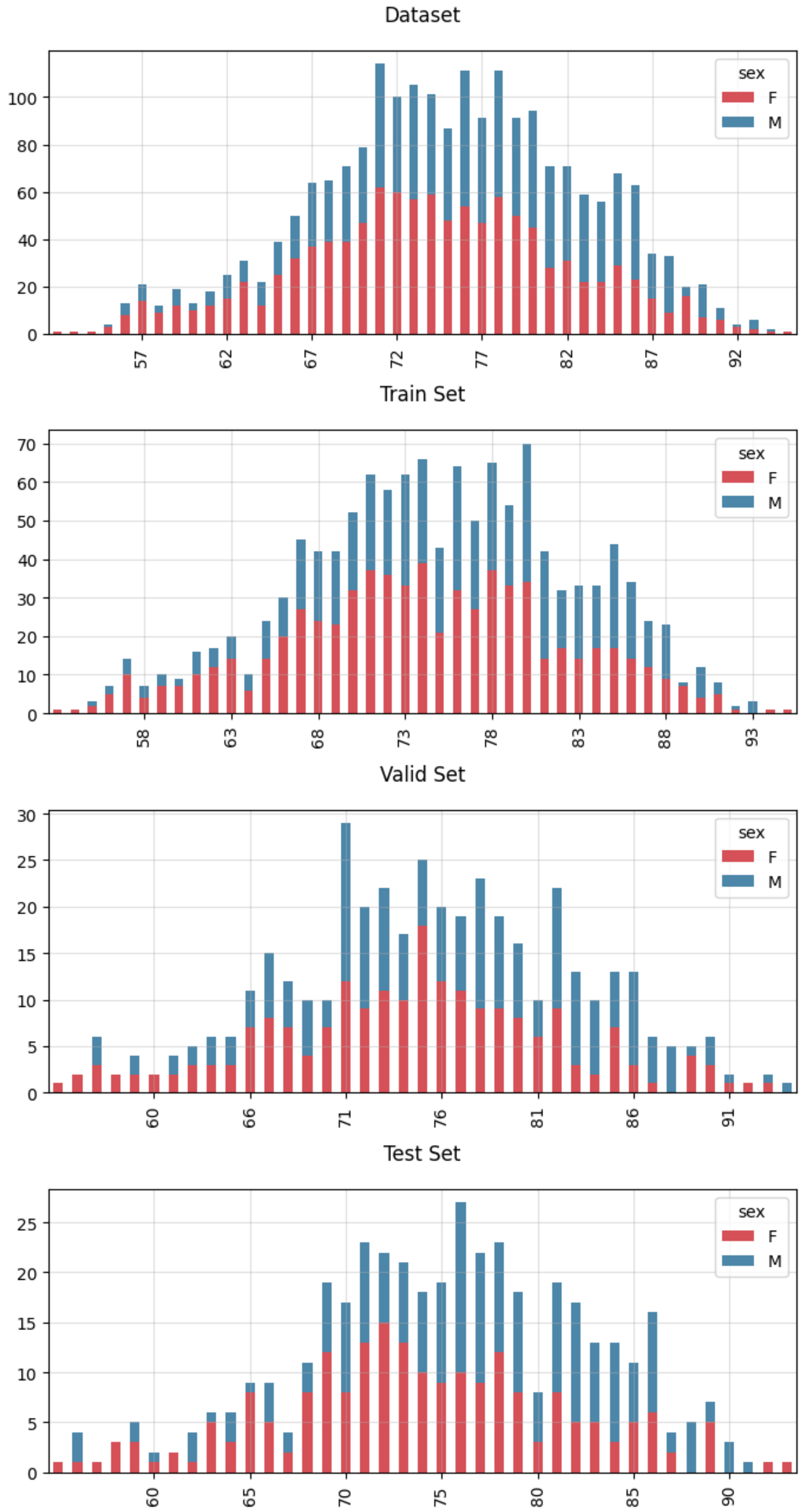


Figure 7: Illustrates the correlation between gender and age within the dataset. The **x-axis** denotes age, while the **y-axis** represents the count. This stacked bar chart displays the counts for each gender, with the count for one gender stacked on top of the other. The **distribution** reveals that up to about 75 years of age, there are more females than males. However, beyond 78 years, there is a slight increase in the number of males compared to females. All sets are almost balanced.

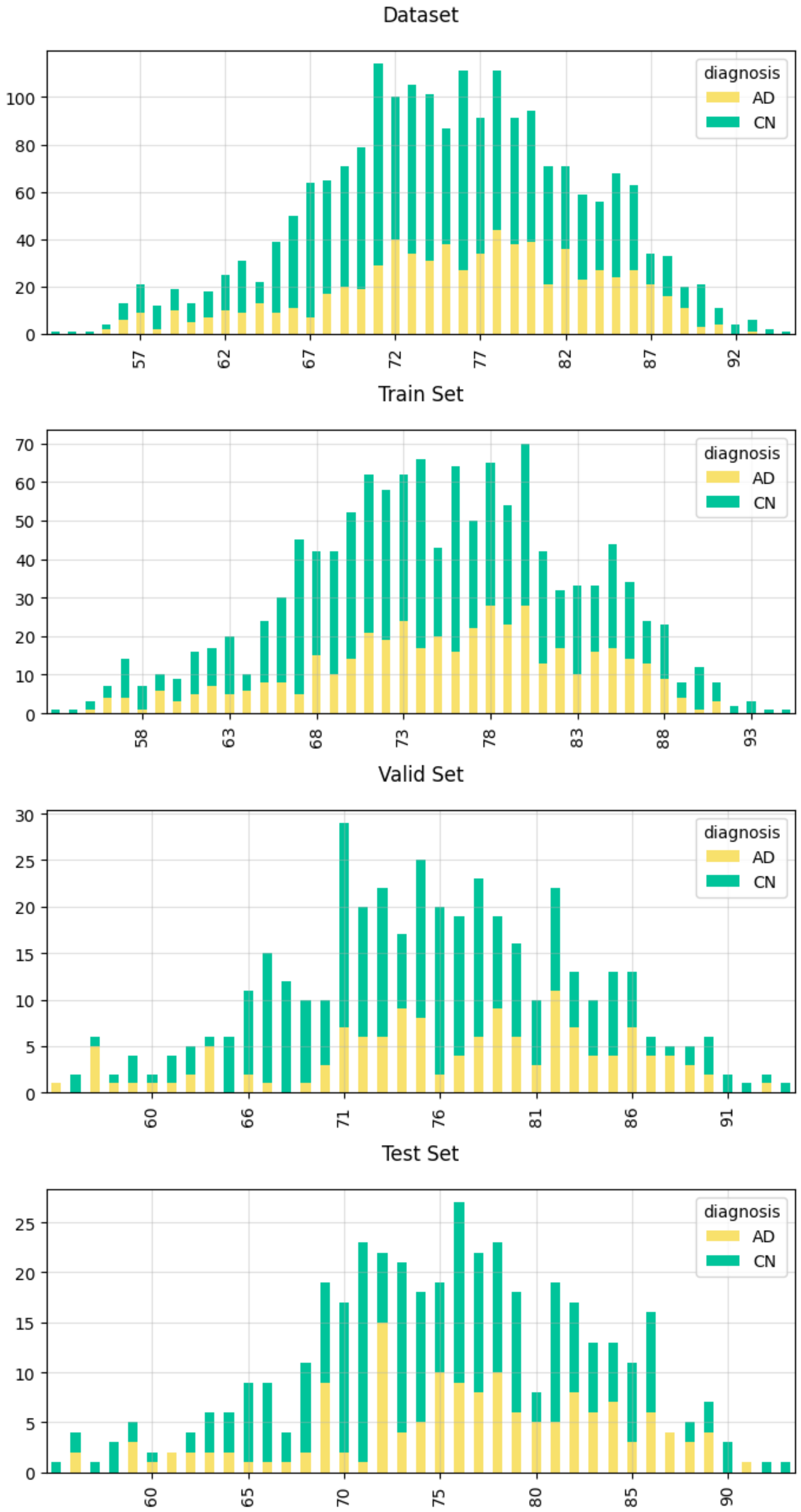


Figure 8: Illustrates the correlation between diagnosis and age within the dataset. The **x-axis** represents age, while the **y-axis** signifies the count. This stacked bar chart displays the counts for each diagnosis class, with the count for one class stacked on top of the other. The **distribution** shows a higher prevalence of cases between the ages of 72 and 82. The Training Set mirrors the overall dataset distribution, while the Test Set and Validation Set do not follow this pattern exactly. However, the Test Set and Validation Set exhibit similar distributions.

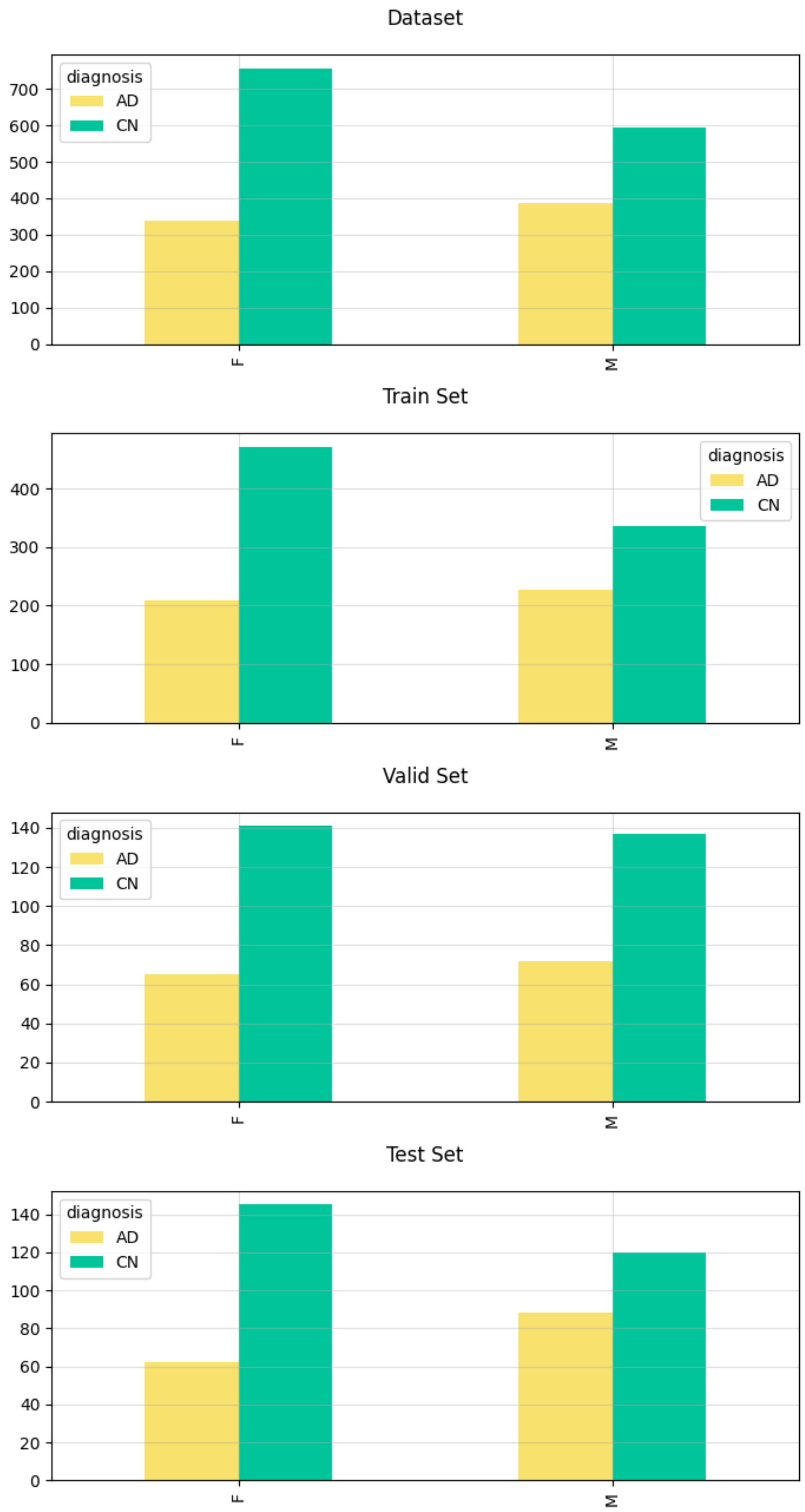


Figure 9: Illustrates the correlation between diagnosis and gender within the dataset. The **x-axis** represents gender, while the **y-axis** indicates the count. This stacked bar chart displays the counts for each diagnosis class, with the count for one class stacked on top of the other. The **distribution** patterns are consistent across all sets mirroring the distribution of the initial dataset.

2.4 Folder Structure of the Dataset

Let's illustrate the folder structure of the dataset `data`. The `raw` directory houses the CSV file of the initial collection downloaded from IDA, which includes unnecessary columns, non-ideal column names, multiple techniques for each scan, and several scans per patient. The `processed` directory contains two files: the first, `dataset.csv`, holds the cleaned and standardized version of the initial dataset filtered to include only the relevant techniques and is limited to two MRI scans per patient. This file is also used subsequently to create a new collection on IDA to download the MRI scans of our final clean dataset; the second, `dataset_nifti.csv`, extends the previously defined dataset by adding a "split" column that identifies whether each row belongs to the training set, validation set, or test set, and a "mri_path" column that contains the path to the "preprocessed.nii.gz" scan for that sample. Finally, the `images` directory will initially include the zipped image files and the NIfTI scans, as previously described. That is the final folder structure:

```
/
├── ...
└── data
    ├── images
    │   └── ADNI
    │       ├── subjects
    │       │   └── SUBJECT_ID
    │       │       └── IMAGE_ID.nii.gz
    │       └── derivatives
    │           └── SUBJECT_ID
    │               └── IMAGE_ID
    │                   ├── skullstrip_mask.nii.gz
    │                   └── preprocessed.nii.gz
    ├── raw
    │   └── collection.csv
    └── processed
        ├── dataset.csv
        └── dataset_nifti.csv
```

3 Proposed Solution

My approach involves a meticulously crafted and efficient methodology for training a deep learning model designed to classify MRI scans to AD or CN. This approach integrates advanced techniques and tools to ensure both precision and robustness in the model's performance.

The training regimen consists of 130 **epochs**, with a **batch size** of 16 samples, a **learning rate** of 0.0001, and a **dropout** rate of 0.2. Dropout is a regularization technique designed to prevent overfitting by randomly omitting a proportion of the network's neurons during training. This helps the model generalize better to unseen data, so helps prevent overfitting.
All the process of training has been executed on my desktop PC with an **Intel Core i7** CPU, **32GB** of RAM, and an **RTX 4070 Super** GPU with **16GB VRAM**.

3.1 Model

Central to this solution is the utilization of the **3D DenseNet** [3], a powerful neural network architecture developed within the MONAI framework. **MONAI (Medical Open Network for AI)** is a specialized library designed for medical imaging tasks, offering state-of-the-art tools and pre-configured models tailored to handle the unique challenges of medical image analysis. The 3D DenseNet, in particular, excels in processing volumetric data by capturing intricate spatial patterns through its densely connected convolutional layers. This architecture is especially well-suited for MRI scans, where detailed three-dimensional feature extraction is crucial.

The primary advantage of the 3D DenseNet lies in its dense connectivity. In this architecture, each layer receives input from all preceding layers, which promotes feature reuse and mitigates the vanishing gradient problem, leading to more efficient and deeper network training. This dense connectivity pattern enhances the network's ability to learn complex patterns and representations from the volumetric MRI data, improving classification accuracy and robustness.
Additionally, the 3D DenseNet benefits from the integration with MONAI, which provides optimized implementations and configurations specifically tuned for medical imaging.

Let's see a formal representation of the model. Given the 3D MRI scan input x, we define the model as the DenseNet denoted as f_θ. The final predicted class probability is obtained as $\hat{y} = f_\theta(x)$. In the output layer of the network, a sigmoid function $\sigma(z)$ is applied to return the probability that x belongs to class 1. Thus, $\hat{y} = f_\theta(x) = P(y = 1|x, \theta)$.

3.2 Loss Function

In this binary classification task, the chosen loss function is the **Binary Cross Entropy Loss** (BCEWithLogitsLoss) from PyTorch. Cross-entropy loss is particularly well-suited for classification problems because it quantifies the difference between the true labels and the predicted probabilities, effectively measuring how well the model's predictions align with the actual outcomes.

$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$

Where $y^{(i)}$ is the true label of the i-th example, $\hat{y}^{(i)}$ is the predicted probability of the i-th example, m is the number of examples, and θ represents the model parameters.

The binary cross entropy loss is **continuous** and **differentiable**, which makes it highly suitable for optimization using gradient descent. The continuity and differentiability ensure that the gradient of the loss function can be computed, allowing the model to update its parameters in the direction that minimizes the loss function $J(\theta)$.

The gradient of the loss function J with respect to a parameter θ_j is given by:

$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right) x_j^{(i)}$

Here, $x_j^{(i)}$ represents the j-th feature of the i-th example. This derivative indicates how the loss function changes with respect to the parameter θ_j, guiding the parameter updates during the training process to minimize the loss.

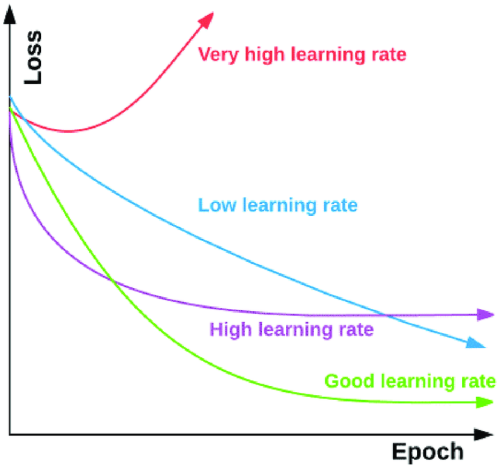
3.3 Optimizer for Learning

In the learning phase of training a neural network, the use of an optimizer is crucial for adjusting the model's parameters to minimize the loss function J. The optimization process involves iteratively updating the model parameters, θ, to reduce the discrepancy between the predicted outputs and the true labels. This iterative process seeks to find the optimal set of parameters θ that minimizes $J(\theta)$, which can be mathematically represented as:

$\theta_{i,j} = \theta - \alpha \frac{\partial J(\theta)}{\partial \theta_{i,j}}$

where α is the learning rate, a hyperparameter that controls the step size during the parameter updates. The learning rate is critical in this context; a value too high can cause the training process to overshoot the minimum, while a value too low can result in slow convergence.

The **AdamW** optimizer is utilized for model training, known for its advantages over traditional Stochastic Gradient Descent (**SGD**). AdamW incorporates weight decay and adaptive learning rates, which enhance its performance. Specifically, AdamW modifies the learning rate of each parameter individually, based on estimates of the first and second moments of the gradients. This results in more efficient convergence and better generalization. Unlike SGD, which updates parameters based solely on the gradient, AdamW's weight decay helps prevent overfitting by penalizing large weights, thus improving model robustness.



4 Model Evaluation

In this chapter, I illustrate the evaluation techniques used for my solution and explain their purpose. Evaluating the performance of a machine learning model is crucial to understanding its effectiveness and reliability in making predictions. I utilised several metrics to assess our binary classification model trained on 3D MRI scans, namely: Accuracy Score, Precision, Recall, F1-Score, and the AUC of the ROC curve. Each metric provides unique insights into the model's performance, addressing different aspects of its predictive capabilities.

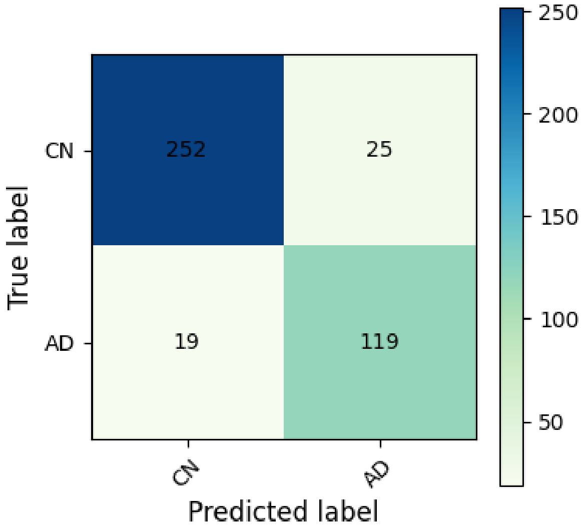
The evaluation metrics indicate a robust model with a strong performance in the binary classification of 3D MRI scans. The accuracy of **89.39%** reflects a high level of correct predictions, while the precision of **82.63%** and recall of **86.23%** demonstrate the model's balanced handling of false positives and false negatives. The F1-Score of **84.39%** further confirms this balance. The AUC of **88.60%** signifies excellent discrimination capability, reinforcing the model's effectiveness in distinguishing between positive and negative cases.

4.1 Accuracy Score

Accuracy Score is the ratio of correctly predicted instances to the total instances. The accuracy score for my model is 89.39%, indicating that nearly 90% of the predictions are correct. However, accuracy alone can be misleading, especially with imbalanced datasets (as in this case) where one class may dominate. It is calculated as Accuracy $= \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$ where TP, TN, FP, and FN represent **true positives**, **true negatives**, **false positives**, and **false negatives**, respectively.

4.2 Confusion Matrix

The Confusion Matrix provides a detailed breakdown of the model's performance by showing the counts of true positive, true negative, false positive, and false negative predictions. This matrix helps visualize the types of errors the model is making. The **diagonal** elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.

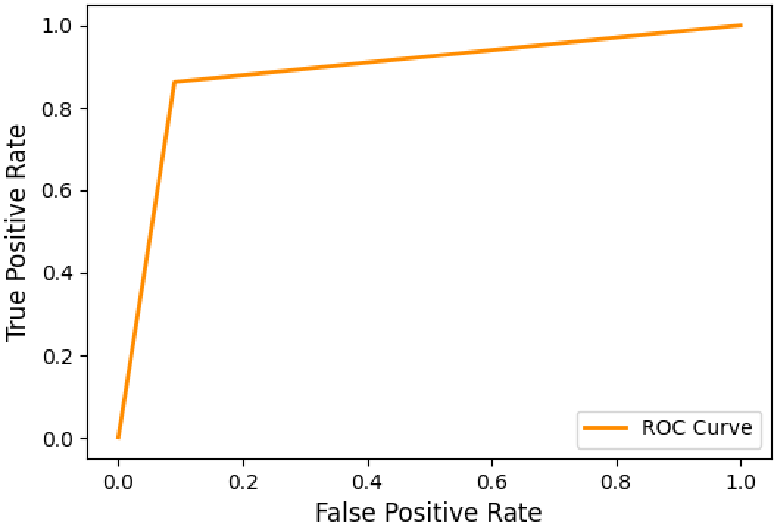


4.3 AUC of the ROC Curve

The ROC curve is a graphical representation used to evaluate the performance of a binary classifier system. The ROC (Receiver Operating Characteristic) curve plots the TP rate against the FP rate. The Area Under the ROC Curve (AUC) provides a single value summarizing the model's performance across all classification thresholds. It is defined as the integral of the ROC curve and can be interpreted as the probability that the model ranks a randomly chosen positive instance higher than a randomly chosen negative one.

The AUC of the ROC curve for my model is 88.60%. A higher AUC indicates better overall performance. The ROC curve and its AUC are useful in evaluating the trade-offs between sensitivity and specificity, which is critical in medical diagnostics.

TP Rate $= \frac{\text{TP}}{\text{TP+FN}}$ and FP Rate $= \frac{\text{FP}}{\text{FP+TN}}$



4.4 Precision

Precision measures the proportion of true positive predictions among all positive predictions made. Precision for my model is 82.63%, meaning that out of all instances predicted as positive, 82.63% were positive. High precision indicates a low false positive rate, which is critical in medical diagnosis to avoid misclassifying healthy patients as having a disease.

Precision $= \frac{\text{TP}}{\text{TP+FP}}$

4.5 Recall

Recall measures the proportion of actual positives correctly identified by the model. My model achieved a recall of 86.23%, indicating that it correctly identifies 86.23% of the actual positive cases. High recall is essential in medical contexts to ensure that as many diseased patients as possible are correctly diagnosed, reducing the risk of missing critical cases.

Recall $= \frac{\text{TP}}{\text{TP+FN}}$

4.6 F1-Score

The F1-Score is the harmonic mean of precision and recall, providing a balance between the two metrics. The F1-Score for my model is 84.39%. It is beneficial when the balance between precision and recall is needed, offering a single measure that considers both FP and FN.

$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

4.7 Comparison with the State-of-the-Art

In evaluating my model against the current state-of-the-art methods in Alzheimer's Disease (AD) diagnosis using 3D MRI scans, I analyzed recent papers to draw comparisons.

One notable study from **2021** by *Hussain et al.* [4] trained a Convolutional Neural Network (CNN) on the same ADNI dataset that I utilized. This study reported an impressive final accuracy score of approximately **98%**. Another significant **2021** study by *AbdulAzeem et al.* [5] used the OASIS dataset to train a CNN for the same binary classification task. The authors reported a final accuracy score of **97.75%** and an AUC of the ROC curve of **99.21%**.

Comparing these results to my model's performance, which achieved an accuracy score of 89.30%, precision of 82.63%, recall of 86.23%, F1-score of 84.39%, and an AUC of 88.60%, it is evident that while my model performs well, there is still room for improvement to reach the benchmarks set by these state-of-the-art approaches.

5 Experiments

Arriving at a satisfactory solution was facilitated significantly by the robust capabilities of the MONAI model employed and the meticulous preprocessing of the MRI scans, which were already of high quality upon download. Despite the inherent quality and robustness of the model and the dataset, it was imperative to **fine-tune** the **hyperparameters** to achieve optimal performance. This process, though potentially complex, was streamlined through careful experimentation. The key hyperparameters considered are the dropout probability, learning rate, weight decay, batch size, and the number of epochs. I needed two experiments to find the optimal hyperparameters.

#	**Dropout Prob**	**Learning Rate**	**Weight Decay**	**Batch Size**	**Num Epochs**
1	2×10^{-1}	1×10^{-3}	1×10^{-5}	16	125
2	2×10^{-1}	1×10^{-4}	1×10^{-5}	16	130

5.1 Experiment 1

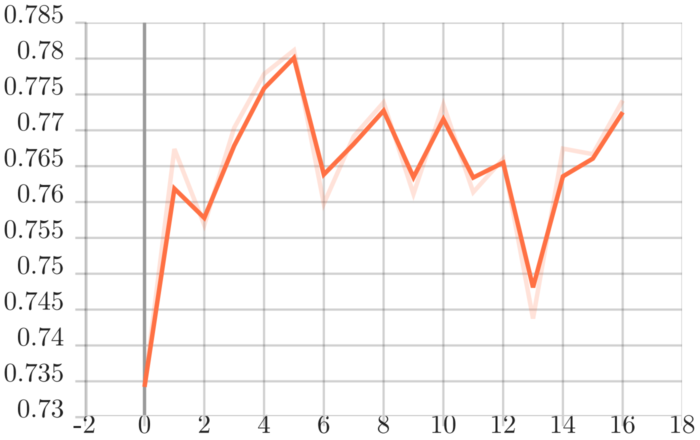


Figure 10: Training Accuracy Curve for the First Experiment.

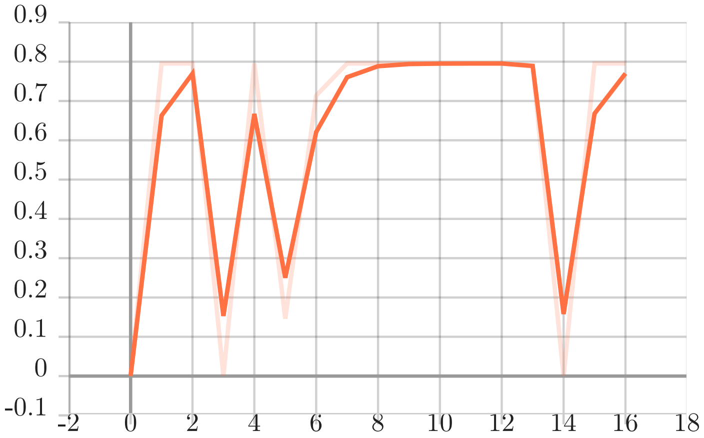


Figure 11: Validation Accuracy Curve for the First Experiment.

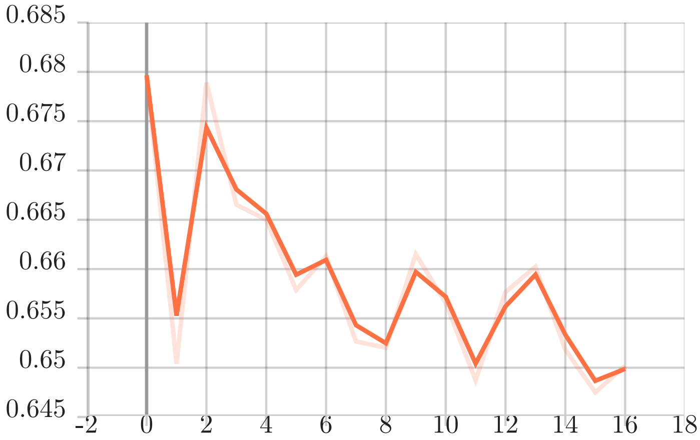


Figure 12: Training Loss Curve for the First Experiment.

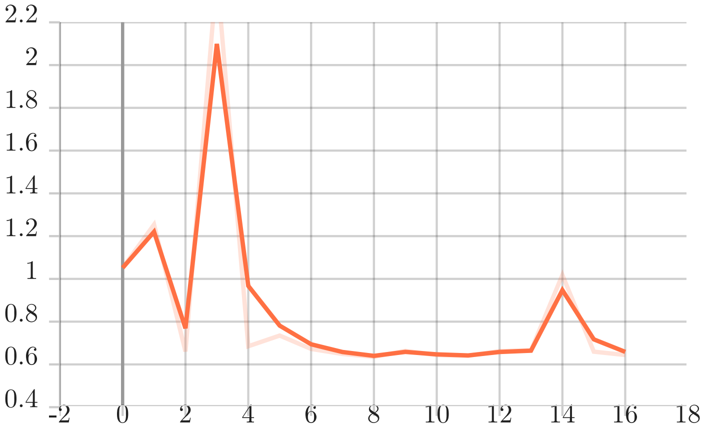


Figure 13: Validation Loss Curve for the First Experiment.

In the first experiment, as indicated by the training and validation loss and accuracy graphs, it was evident that the model was indeed learning. However, the curves were excessive **irregular**, suggesting an unstable training process. This irregularity is often indicative of a learning rate that is too high, causing the model parameters to oscillate rather than converge smoothly. Given the limited computational resources available, I opted for **early stopping** after **16 epochs**. Early stopping is a regularization technique used to prevent overfitting by halting the training process when the performance on the validation set starts to deteriorate.

By observing the patterns in the loss and accuracy graphs, I was able to make informed decisions about modifying the learning rate. This adjustment was crucial for improving the model's performance, ensuring that the learning process was both effective and efficient.

5.2 Experiment 2

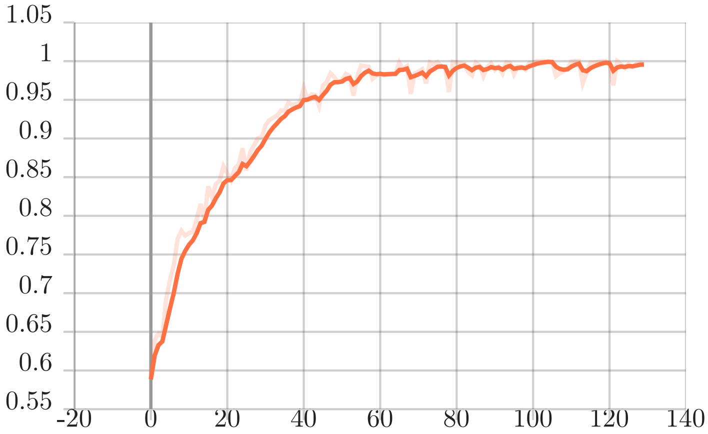


Figure 14: Training Accuracy Curve for the Second Experiment.

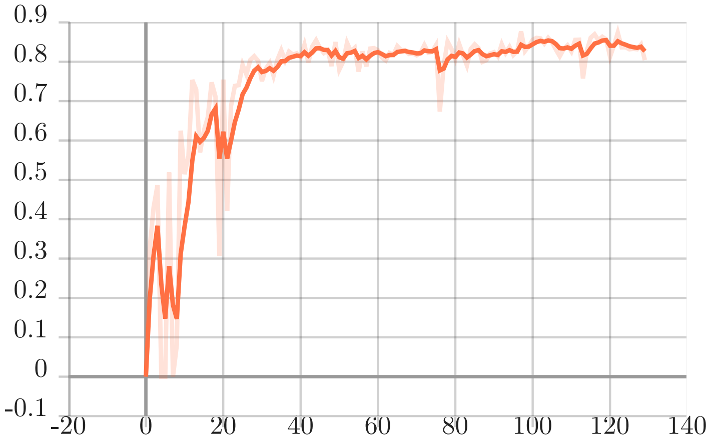


Figure 15: Validation Accuracy Curve for the Second Experiment.

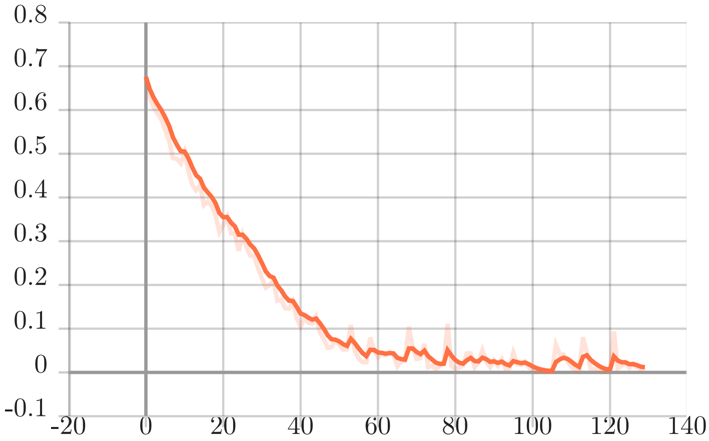


Figure 16: Training Loss Curve for the Second Experiment.

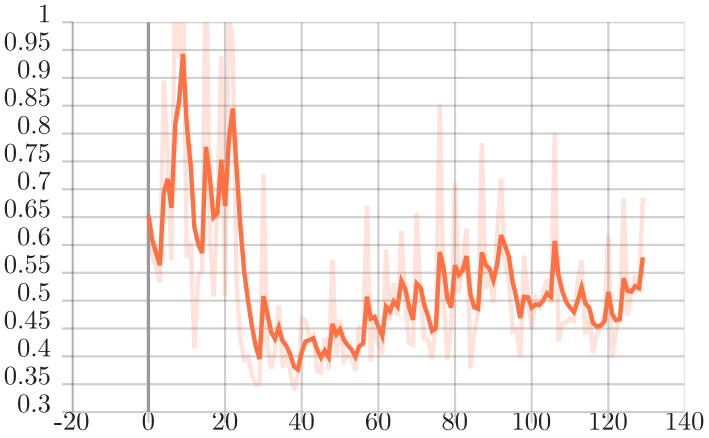


Figure 17: Validation Loss Curve for the Second Experiment.

In the second experiment, as evidenced by the training loss and the training and validation accuracy graphs, the model was learning with **significantly less irregularity** in the curves. This improvement indicated a more stable training process compared to the first experiment. Consequently, I allowed the model to continue its training. The final results, as discussed in the evaluation section, are highly satisfactory. So, I chose this model as the final model.

Although the validation loss graph remained **irregular**, this can be attributed to several factors. First, the use of small batch sizes inherently introduces more noise into the training process, which can lead to fluctuations in the validation loss. Second, the accuracy was calculated using the F1 score, a metric that is particularly sensitive to the balance between precision and recall, which can also contribute to variability. Lastly, the complexity of the task itself, involving the classification of 3D MRI scans, adds another layer of difficulty.
Despite these irregularities, my supervisor, PhD student Lemuel Puglisi and I concluded that the results were acceptable. With more time and resources, I **would** have invested further efforts into refining hyperparameters and potentially smoothing out the validation loss curve.

6 Source Code

The code for my project is organized into files and directories in an orderly fashion. This adherence to **Software Engineering principles** ensures that the project maintains high readability, maintainability, and reusability. The structure, excluding files such as *LICENSE*, *README*, and directories for `docs` and `reports`, is represented as follows:

```
/
├── ...
├── requirements.txt
├── data/
│   └── ...
├── models/
│   └── ...
├── notebooks/
│   ├── 01-dataset-analysis.ipynb
│   ├── 02-train.ipynb
│   ├── 03-test.ipynb
│   └── 04-demo.ipynb
└── src/
    ├── data/
    │   ├── dataset.py
    │   └── transforms.py
    └── utils/
        ├── log.py
        ├── meter.py
        ├── model.py
        └── visualisation.py
```

The needed **dependencies** are in the `requirements.txt` file and can be easily installed by running the command `pip install -r requirements.txt` in the terminal open in the project root directory.

6.1 Jupyter Notebooks

There are four Jupyter notebooks divided into two **categories**: those dedicated entirely to the dataset and those dedicated to the model. The first notebook `01-dataset-analysis.ipynb`, analyses the dataset and its three subsets to examine distributions and balances.

The next two notebooks, `02-train.ipynb` and `03-test.ipynb`, are used for **training** and **testing** the model, respectively. The final notebook, `04-demo.ipynb`, is utilized for making **inferences**. During the training phase, the notebook `02-train.ipynb` continually saves the best model in the `/models/` directory. Both testing and demo notebooks, `03-test.ipynb` and `05-demo.ipynb`, respectively, access this saved model file to carry out their functions.

Users should only interact with these notebooks and no other code files. The files in the `/src/` directory are merely modules (classes and functions) called within the notebooks to keep them clean and readable. The order of use by the user depends on the specific use case:

- If the user already **owns** the **dataset** (both CSV and MRI scans), but doesn't want to use my ready-to-use model present in the `/model/` directory, they can run the notebooks `02-train.ipynb`, `03-test.ipynb`, and `04-demo.ipynb`.
- If the user already **owns** the **dataset** (both CSV and MRI scans) and wants to use the **model** present in the `/model/` directory, they can run the notebooks `03-test.ipynb` and `04-demo.ipynb` in the order they like.

6.2 Python Source Code

Inside the `/src/` directory, there are two key subdirectories: `data/` and `utils/`. Each contains Python source files with modules used within the notebooks.

These organised modules significantly contribute to the project by ensuring that the code remains modular, readable, and maintainable. This structure also facilitates easier debugging and extension of the project, allowing for scalable development and experimentation.

The `data/` directory includes files essential for managing the data during training, testing, and partially during inference in the demo. Specifically, these files handle tasks such as data loading, augmentation, and applying transformations (e.g. crop, normalization).

The `utils/` directory contains files with classes and functions designed to streamline the code in the notebooks across the three main phases: training, testing, and inference. These utility modules include common functionalities such as metric calculations, logging, visualisation, and other helper functions that contribute to notebooks' overall readability and maintainability.

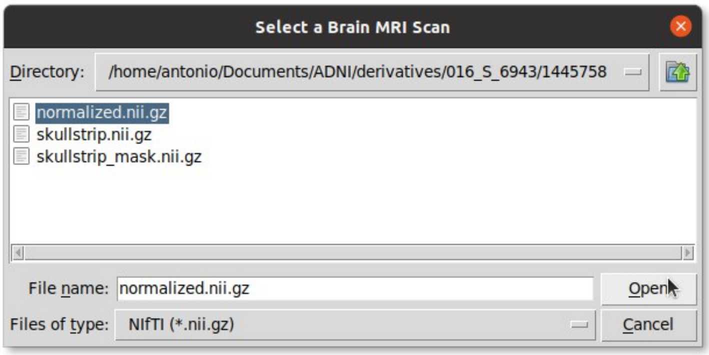
- `data/dataset.py`: The **BrainMriDataset** class defined in this file, allows easy access to MRI images and their corresponding labels, making it straightforward to integrate with PyTorch's *DataLoader* for batch processing during training and testing. The CN class is mapped to 0 (Negative for Alzheimer's), and the class AD is mapped to 1 (Positive for Alzheimer's).
- `data/transforms.py`: The **Transforms** class provides a set of static methods for applying transformations to the MRI data. These transformations standardize the MRI images, ensuring consistency in size, spacing, and intensity, which is critical for training and testing robust machine learning models and making inferences on them.
- `utils/log.py`: This file contains a function designed to log the loss and accuracy (F1-score) of both training and validation to **TensorBoard**.
- `utils/meter.py`: This file defines a class that computes average metrics over multiple samples. The class is used to track loss and accuracy (F1-score) metrics across epochs. Although the

class itself does not specify how metrics are accumulated, it is utilized during the training phase to store values for each batch, averaging them over an epoch. At the end of each epoch, the class is reset to ensure accurate tracking for the subsequent epoch.

- `utils/model.py`: This file includes functions for saving and loading the model, as well as calculating loss and performance metrics. Ensures that the best-performing model and its optimizer state can be saved for future use or evaluation. Facilitates the retrieval and deployment of the best-performing model for inference or further training.
- `utils/visualisation.py`: This file contains a set of functions designed for visualizing various aspects of the model's performance and MRI scans. It provides the functions to plot the ROC Curve and the Confusion Matrix for the evaluation and to display the central slices of an MRI scan from different perspectives (axial, coronal, sagittal) for the demo.

7 Demo

The project includes a demo for model inference, accessible via a Jupyter notebook. This notebook provides a user-friendly interface for testing the trained model with new MRI scans.
Once the notebook is executed, the user is prompted to select a NIfTI file containing the MRI scan they wish to classify. Upon successful loading of the file, the model performs inference and displays the predicted class. This allows for an interactive and practical demonstration of the model's capabilities, making it easy to assess its performance with new, unseen data.



Select a Brain MRI Scan

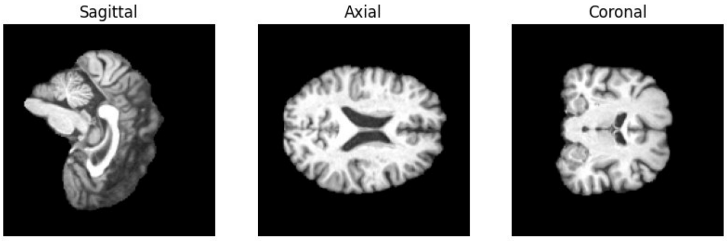
Directory: /home/antonio/Documents/ADNI/derivatives/016_S_6943/1445758

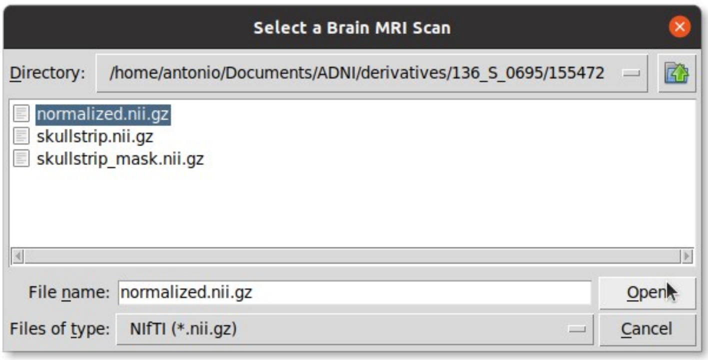
normalized.nii.gz
skullstrip.nii.gz
skullstrip_mask.nii.gz

File name: normalized.nii.gz

Files of type: NIfTI (*.nii.gz)

Predicted [CN]





Select a Brain MRI Scan

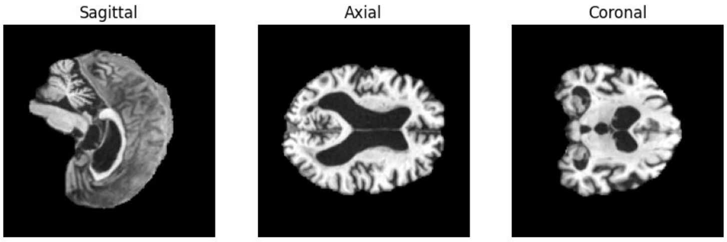
Directory: /home/antonio/Documents/ADNI/derivatives/136_S_0695/155472

normalized.nii.gz
skullstrip.nii.gz
skullstrip_mask.nii.gz

File name: normalized.nii.gz

Files of type: NIfTI (*.nii.gz)

Predicted [AD]



8 Conclusions

Reflecting on this project, I have gained valuable insights and learned several important lessons. The **most significant lesson** has been the transition from theoretical knowledge acquired in lectures to practical implementation in a real-world context. Machine learning in practice is not merely about experimenting randomly but involves a disciplined approach to **reasoning** and selecting appropriate functions and hyperparameters. This project has honed my ability to **think critically** and **make informed decisions** at various stages of the model development process.

Engaging deeply in the field of medical imaging has been both intellectually stimulating and **socially meaningful**. This area of study not only presents fascinating mathematical and computational challenges but also holds the potential for significant societal impact. Through this project, I have had the opportunity to explore advanced topics in medical imaging, which are both innovative and beneficial to public health. I have delved into new and exciting subjects, such as the intricacies of MRI preprocessing tools, which were essential for preparing the data used in my model. This exploration has expanded my understanding of the domain and equipped me with practical skills in handling complex medical imaging datasets.

From a practical standpoint, this project has also allowed me to develop and refine my skills in designing and implementing a machine learning project. Also, I gained hands-on experience with shell scripting for the preprocessing of MRI data, further broadening my technical expertise.

Reflecting on potential improvements, with additional time and resources, I would have continued to experiment with varying **hyperparameters**. One significant enhancement would involve incorporating **segmentation** techniques to guide the model explicitly on what and where to focus. In its current form, the model may be learning patterns based on the overall intensity distribution in the MRI scans, potentially inferring diagnoses from the presence of "black" areas indicative of brain atrophy. This approach, while functional, lacks specificity and robustness, particularly if we were to introduce a third class, such as **Mild Cognitive Impairment (MCI)**. The model might struggle with this additional complexity without targeted guidance.
Implementing segmentation would allow the model to concentrate on specific brain regions critical for diagnosing conditions like Alzheimer's Disease (AD). For instance, prior knowledge indicates that the hippocampus is a key region affected by AD. By segmenting the MRI scans to highlight the hippocampus and other relevant areas, the model could learn more precise and reliable features, improving its diagnostic accuracy and robustness across different classes. Furthermore, segmentation would enhance the interpretability of the model's decisions, providing clearer insights into which brain regions contribute to its predictions. This could be particularly beneficial in a clinical setting, offering more transparent and actionable information to healthcare professionals.

References

[1] Juntu, J., Sijbers, J., Van Dyck, D., Gielen, J. (2005). Bias Field Correction for MRI Images. In: Kurzyński, M., Puchała, E., Woźniak, M., żołnierek, A. (eds) Computer Recognition Systems. Advances in Soft Computing, vol 30. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-32390-2_64

[2] Isensee F, Schell M, Tursunova I, Brugnara G, Bonekamp D, Neuberger U, Wick A, Schlemmer HP, Heiland S, Wick W, Bendszus M, Maier-Hein KH, Kickingereder P. Automated brain extraction of multi-sequence MRI using artificial neural networks. Hum Brain Mapp. 2019; 1–13. https://doi.org/10.1002/hbm.24750

[3] `https://docs.monai.io/en/stable/networks.html#densenet`

[4] E. Hussain, M. Hasan, S. Z. Hassan, T. Hassan Azmi, M. A. Rahman and M. Zavid Parvez, "Deep Learning Based Binary Classification for Alzheimer's Disease Detection using Brain MRI Images," 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), Kristiansand, Norway, 2020, pp. 1115-1120, doi: 10.1109/ICIEA48937.2020.9248213.

[5] AbdulAzeem, Y., Bahgat, W.M. & Badawy, M. A CNN based framework for classification of Alzheimer's disease. Neural Comput & Applic 33, 10415–10428 (2021). https://doi.org/10.1007/s00521-021-05799-w