Master Ingénierie en Intelligence Artificielle (I2A)
Advanced Programming Tools for AI

Author: Emmanuel Jonathan EBONGOM MANYOL

End-of-Semester project

Research question: How can AI help identify a specific disease based on Brain MRI images

Objective: This end-of-semester project aims to design and develop a machine learning model using TensorFlow or Scikit-learn and by covering the entire development cycle of a machine learning project, from data processing to model training and evaluation. We will work with TensorFlow on **Brain MRI Image Classification with Explainability and Environmental Impact Evaluation**

Keywords: Computer vision, health, MRI images, deep learning

Table of Contents

1 Introduction

This project aims to build a deep learning model capable of analyzing brain MRI images, to detect and classify specific brain conditions. Here, three categories will be considered: **healthy_brain, tumor_brain, and Alzheimer_brain**. We will focus on detecting the presence of brain tumors and Alzheimer's disease.

The classification model is built using TensorFlow and includes an important component of model explainability through semantic segmentation. This helps interpret the regions in the MRI images that influence the model's decision. Furthermore, we explore the environmental impact of training the model by estimating its carbon footprint.

2 Project Initialization

2.2. Data sources

For this project, we use MRI images from the following datasets:

- **Brain Tumor Detection**: Brain Tumor Dataset 1, Brain Tumor Dataset 2
- **Alzheimer Detection**: Alzheimer MRI Dataset

2.3. Directory Structure

The project is organized with the following structure:

```
project/
├── data/                      # For processed datasets
├── datasets_original_cleaned/ # For cleaned downloaded original datasets
├── notebooks/                 # Initial exploratory analysis
├── models/                    # Trained models saving place
├── reports/                   # Charts and analyses
├── src/                       # Source code (preparation, training, etc.)
└── README.md                  # Instructions
```

Figure 1: Initial directory structure

2.4. Datasets Organization

The cleaned datasets are structured as follows:

- **Brain Tumor Detection Dataset**:

    - no/: Images of healthy brains.
    - yes/: Images of brains with tumors.
- **Brain Tumor Dataset**:
    - Brain Tumor/: Images of brain tumors.
    - Healthy/: Images of healthy brains.
- **Augmented Alzheimer Dataset**:
    - Mild Demented/: Mild dementia cases.
    - Moderate Demented/: Moderate dementia cases.
    - Non Demented/: No signs of dementia.
    - Very Mild Demented/: Very mild dementia.

Final Data Organization: The final dataset, organized for analysis and training, consists of:

- **Brain Tumor Detection**: A combination of images from both tumor and healthy brain categories.
- **Alzheimer Detection**: A combination of images from both demented and non-demented brain categories.

3 Pre-processing and Exploratory Data Analysis

3.1 Pre-processing and Normalization of Images

- **Rescaling**: The images are normalized to ensure consistent values across all samples. This is a necessary step for training deep learning models efficiently.

3.2 Dataset Overview

To understand the distribution of classes, we explored and visualized the dataset. We perform the following:

- Check how many images belong to each class (healthy brain, brain tumor, non-demented brain, demented brain).

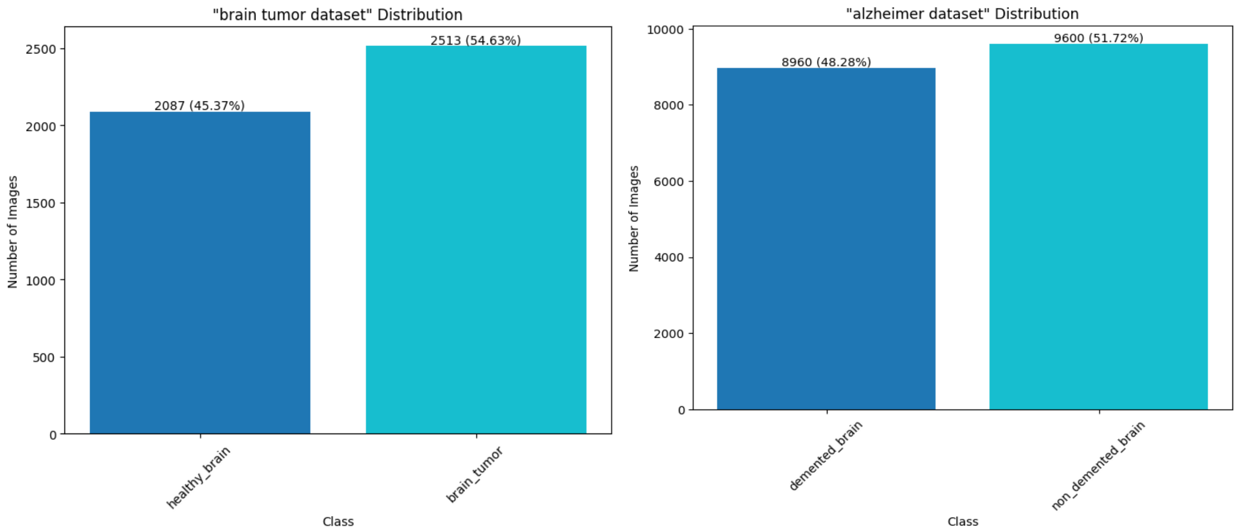


Figure 2: Class data distributions

- Visualize a sample of images from the "healthy brain" and "brain tumor" classes.

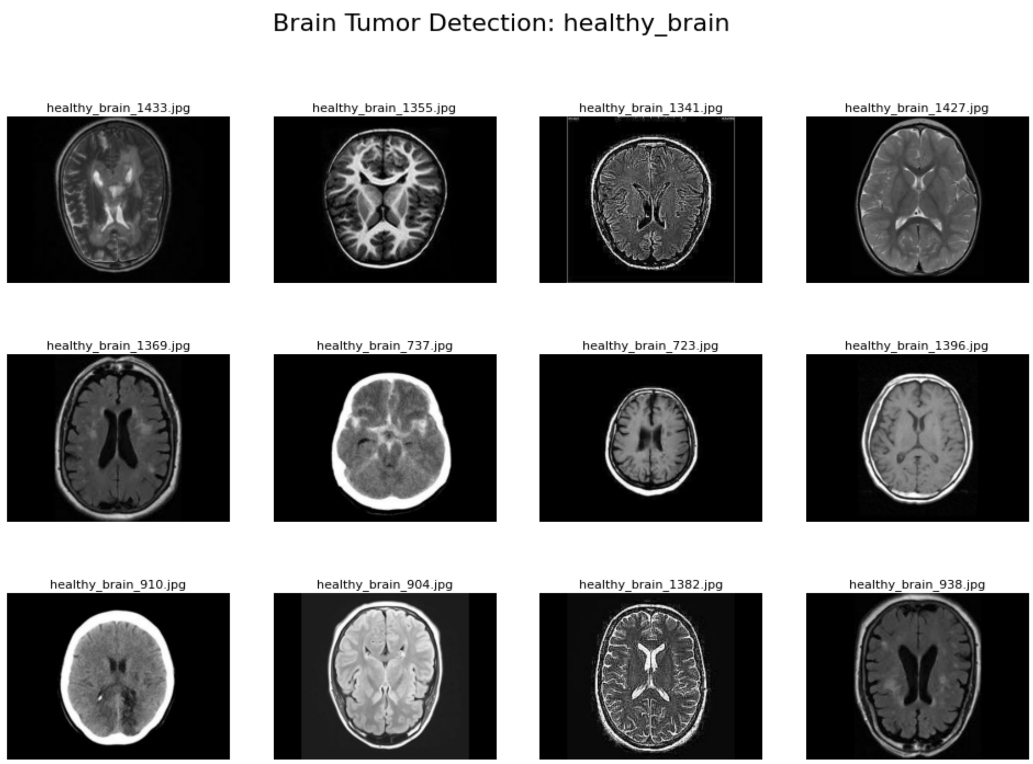


Figure 3: Healthy brain MRI image samples

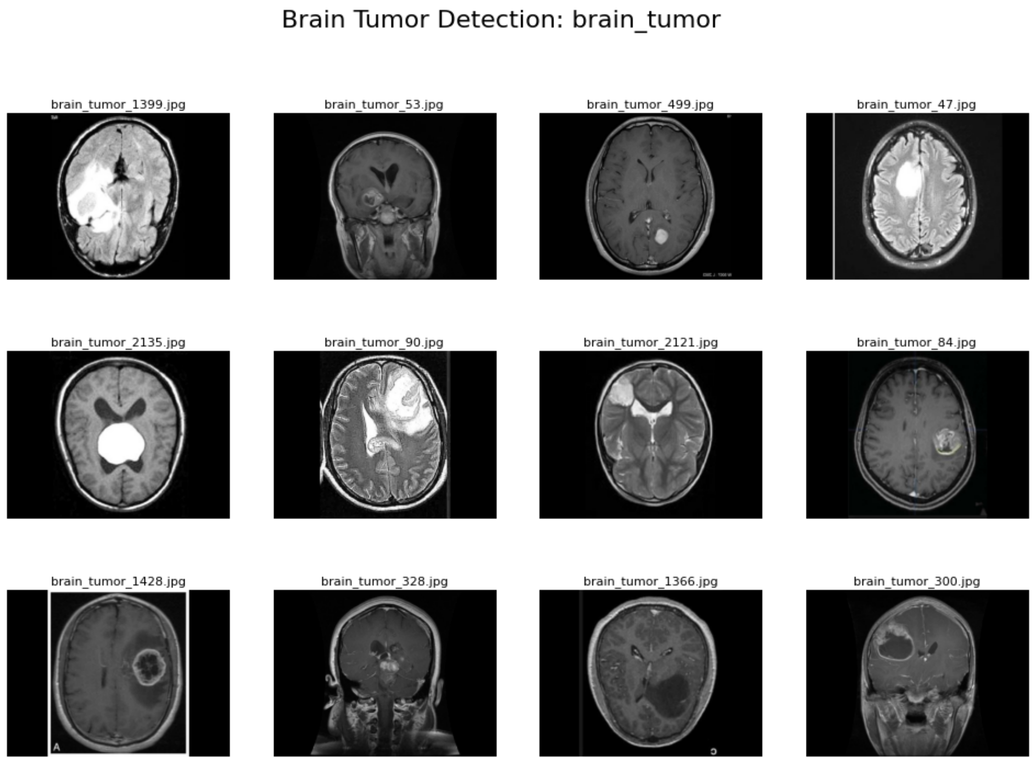
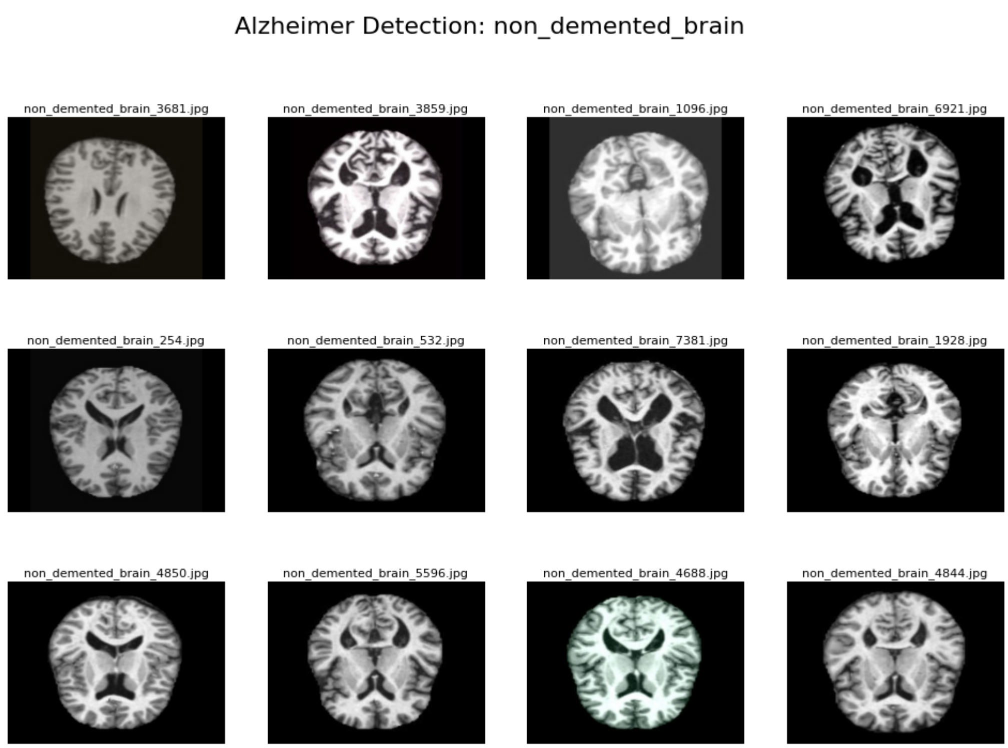


Figure 4: Tumored brain MRI image samples

- Visualize a sample from the "non-demented brain" and "demented brain" categories.



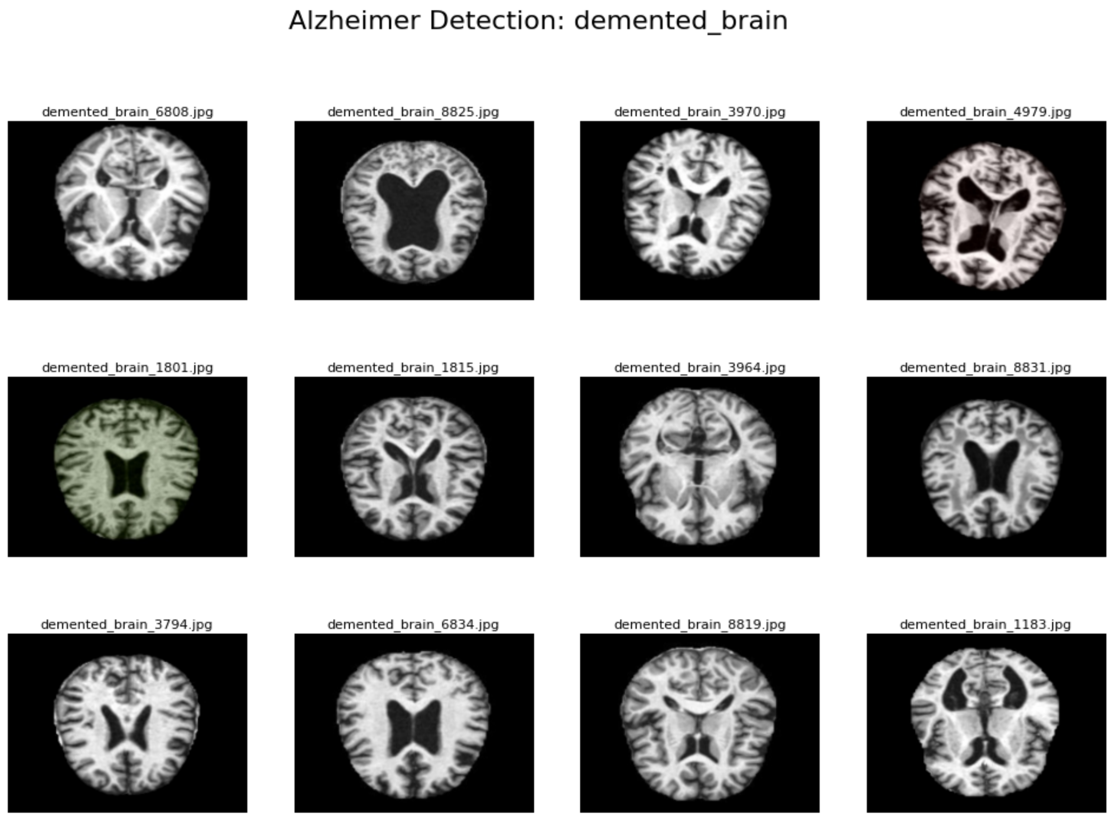
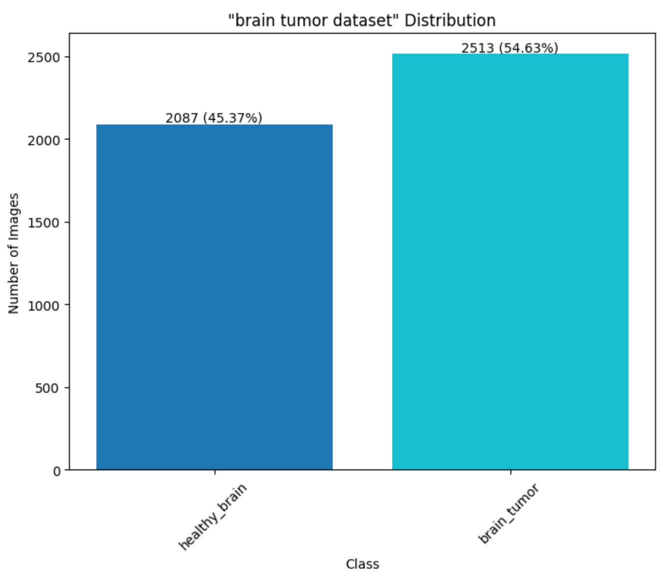


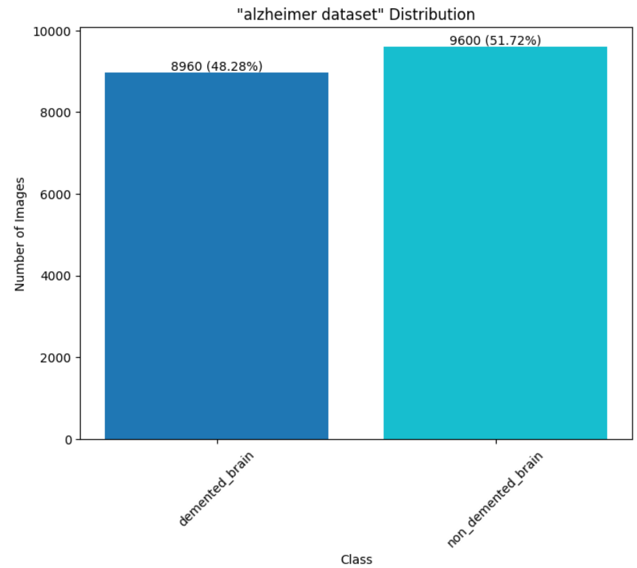
Figure 5: Alzheimer MIR images samples

3.3 Distribution and Class Imbalance state



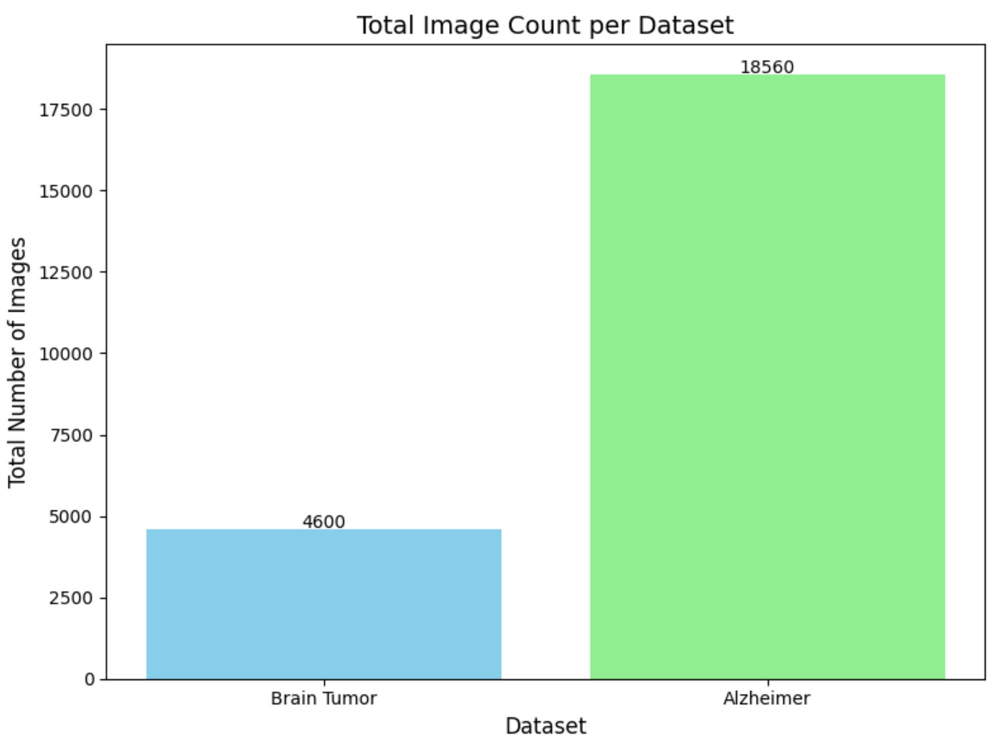
- **Brain Tumor Dataset**:
  - **Total images**: 4600
    - **2513** images labeled **"brain tumor"**
    - **2087** images labeled **"healthy brain."**

- **Alzheimer Dataset**:
  - Total images: 18560
    - **8960** images of "demented brains"
    - **9600** images of "non-demented brains."

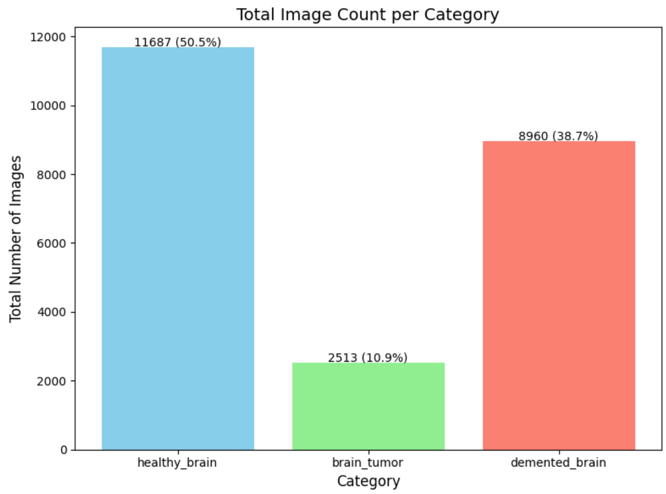


3.4 Data Imbalance:

- There is a noticeable class imbalance in the **Brain Tumor** dataset. However, the categories within each dataset (healthy vs. tumor, non-demented vs. demented) are balanced.
- To address this, random sampling will be used to balance the dataset for model training by ensuring that the number of healthy and Alzheimer's brain images matches the number of brain tumor images.



4 Final Dataset Construction



To ensure that the dataset has balanced classes, we randomly sample from the healthy_brain and non_demented_brain classes to match the tumor_brain class count, which is the smallest. The final dataset will have the following class distribution:

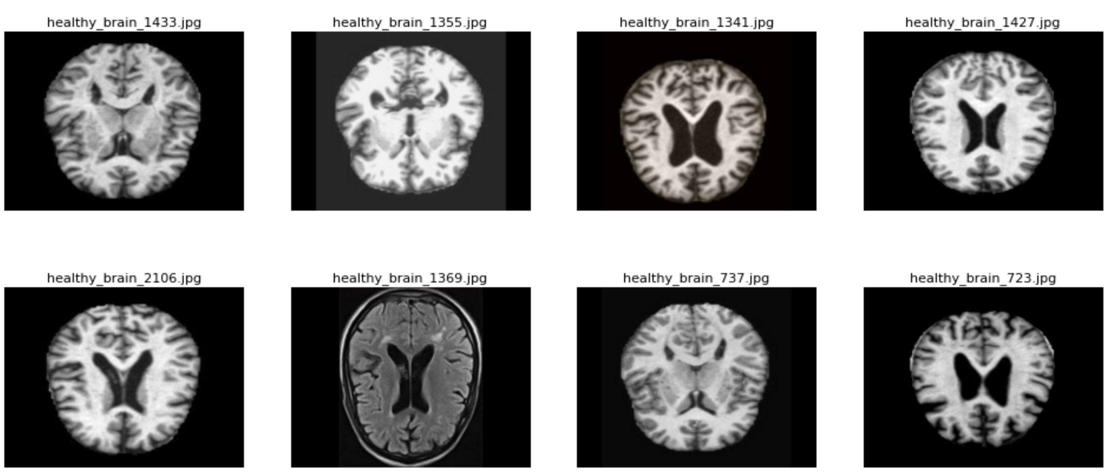
- **healthy_brain**: 2513 images
- **tumor_brain**: 2513 images
- **alzheimer_brain**: 2513 images

This balancing step is crucial for ensuring the model isn't biased toward the dominant class and can effectively learn to detect all three conditions.

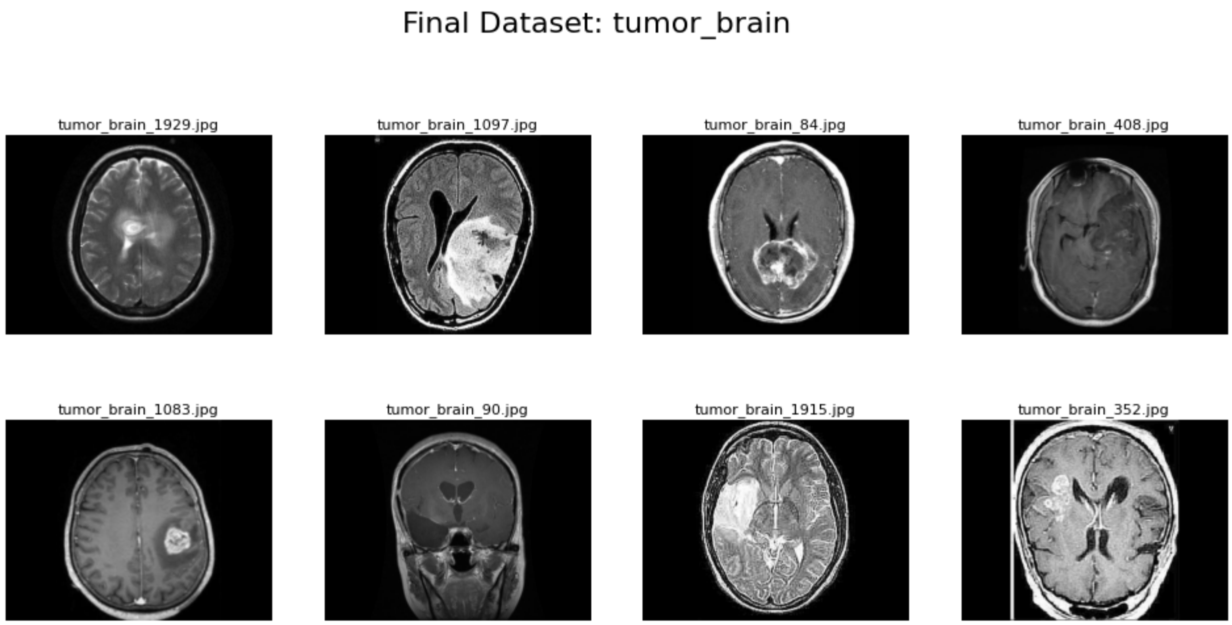
4.1 Final Dataset Preview

- **Healthy Brain**: A preview of images from the **healthy_brain** class.

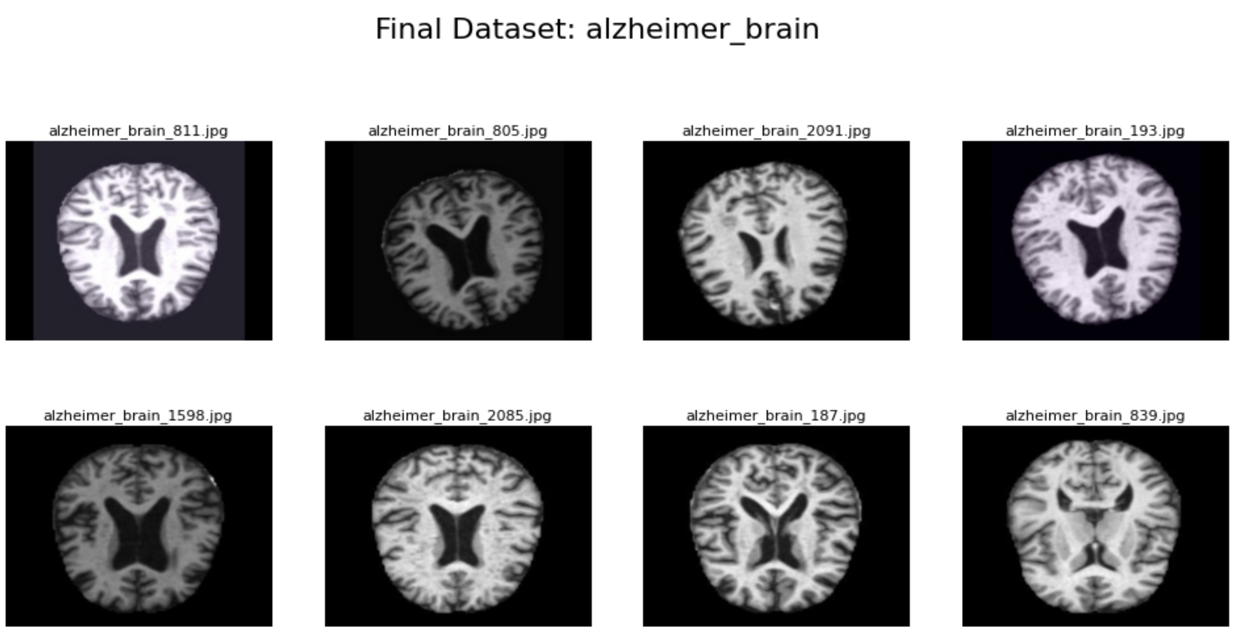
Final Dataset: healthy_brain



- **Tumor Brain**: A preview of images from the **tumor_brain** class.



- **Alzheimer Brain**: A preview of images from the **alzheimer_brain** class.



5 Data Preparation and Model Building

5.1 Data Preparation: Final dataset structure

In the previous notebook, we worked on constructing the final dataset for our model, ensuring that it contains three classes: **`healthy_brain`**, **`tumor_brain`**, and **`alzheimer_brain`**. This final dataset, as shown below, is ready to be used for the modeling phase.

```
final_dataset/
├── healthy_brain/
│   ├── healthy_brain_1.jpg
│   ├── healthy_brain_2.jpg
│   ├── healthy_brain_3.jpg
│   └── ...
├── tumor_brain/
│   ├── tumor_brain_1.jpg
│   ├── tumor_brain_2.jpg
│   ├── tumor_brain_3.jpg
│   └── ...
└── alzheimer_brain/
    ├── alzheimer_brain_1.jpg
    ├── alzheimer_brain_2.jpg
    ├── alzheimer_brain_3.jpg
    └── ...
```

Figure 6: Final Dataset structure

This dataset serves as the foundation for training our model to classify MRI images into the three aforementioned classes.

Next, we split the data into **train, validation, and test sets**. The shapes of these sets are as shown next:

- **Training data shape**: (1920, 512, 512, 3)
- **Training labels shape**: (1920,)
- **Validation data shape**: (480, 512, 512, 3)
- **Validation labels shape**: (480,)
- **Test data shape**: (150, 512, 512, 3)
- **Test data shape**: (150,)

Training and Validation sets have been made using the scikit-learn **train_test_split function** with a ratio of 80%-20%. The Test set is made of completely unseen images put aside expressly for this task (50 images per class).

5.2 Model Architecture

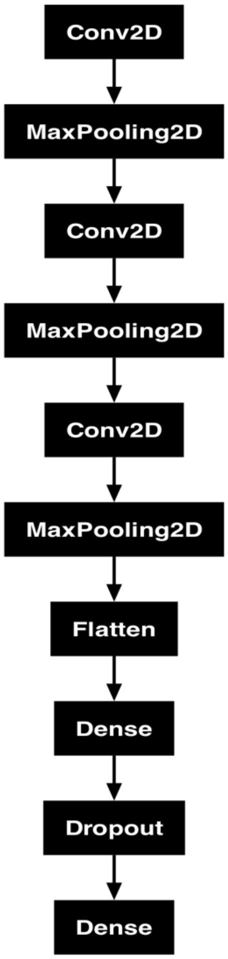


Figure 7: Model architecture

The model was built using TensorFlow's Keras API. We designed a Convolutional Neural Network (CNN) with three convolutional layers followed by max-pooling layers. The final layer uses softmax activation for multi-class classification.

5.2.1 CNN Architecture

- **Input**: 512x512 pixel MRI images.
- **Output**: Three possible classes for each image.
- Use three convolutional layers followed by max-pooling layers.
- Flatten the output and connect it to a fully connected layer with 128 units.
- Final layer uses softmax activation to output the class predictions.
- **Optimizer**: Adam optimizer
- **Learning_rate**: 0.001
- **Loss**: categorical crossentropy
- **Metrics**: Accuracy

5.2.2 Model architecture component explanation

- **Conv2D Layers:** These layers are responsible for extracting features from the MRI images.
- **MaxPooling2D Layers:** Reduce the spatial dimensions of the feature maps, helping the model generalize better.
- **Dropout Layer:** Regularizes the model to prevent overfitting by randomly dropping units during training.
- **Dense Layers:** Fully connected layers that make the final classification based on the features extracted by the convolutional layers.

This CNN architecture is designed to handle image classification tasks efficiently, learning to detect patterns in MRI scans that distinguish between the different brain states.

5.2.3 Model Training

We trained the model using the training dataset and monitored its performance. The following steps were taken:

1. **Preprocessing:** We normalized the images to the range [0, 1] and one-hot encoded the labels.

2. **Training Process:** We trained the model for 20 epochs with a batch size of 32.

3. **Regularization:** Dropout was used to prevent overfitting.

The model training was monitored with loss and accuracy curves to ensure the model was converging and learning effectively. The model stops at 8 epochs since the early_stopping strategy was defined in case performance was not improving:

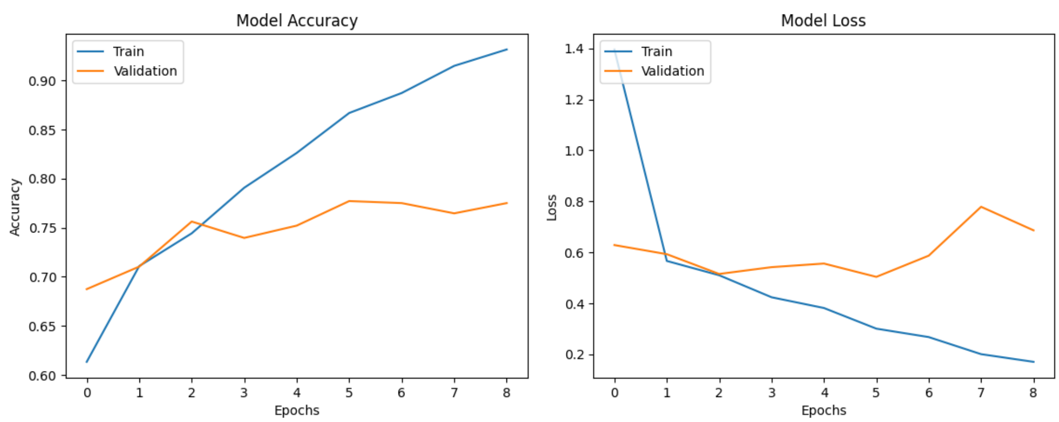


Figure 8: Accuracy & Loss learning curves

5.2.4 Model Evaluation

After training, we evaluated the model on the validation dataset using various metrics such as accuracy, F3-score, and AUC. The model achieved the following performance metrics:

- **Accuracy (on validation set):** 80.67%
- **F3 Score (weighted):** 80.60%
- **Confusion Matrix:**

- **Classification report**:

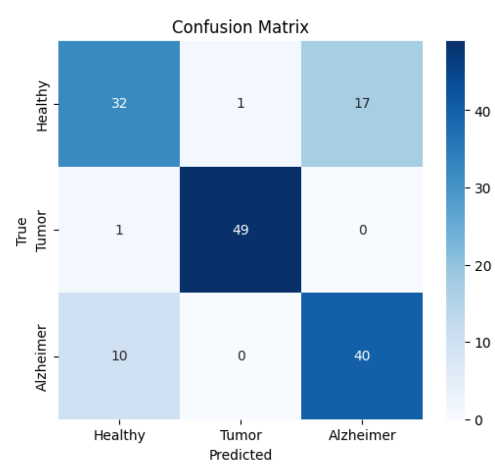


Figure 9: Confusion matrix results on validation set

	precision	recall	f1-score	support
0	0.74	0.64	0.69	50
1	0.98	0.98	0.98	50
2	0.70	0.80	0.75	50
accuracy			0.81	150
macro avg	0.81	0.81	0.81	150
weighted avg	0.81	0.81	0.81	150

Figure 10: Classification report on validation set

- **ROC-AUC Curve**: Evaluate model performance across all classification thresholds.

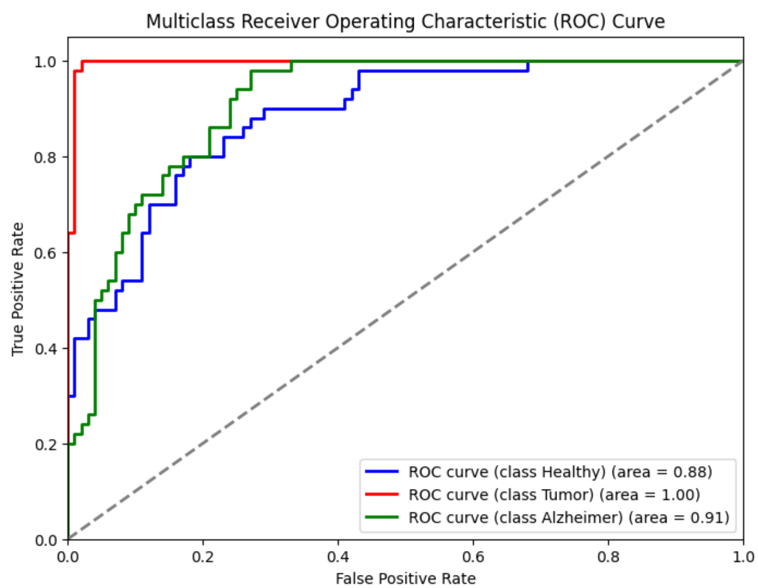


Figure 11: ROC-AUC plot on model performance per classes

This indicates that the model performs relatively well, but there is room for improvement, especially in distinguishing **healthy_brain** and **alzheimer_brain** images.

5.3 Semantic Segmentation for Explainability

To interpret the model's decisions, we applied **Grad-CAM** (Gradient-weighted Class Activation Mapping), a technique for visualizing the regions of an image that influence the model's classification (contribute the most to the model's decision-making process.).

5.3.1 Grad-CAM Visualization

Grad-CAM generates heatmaps that overlay the original image, highlighting the areas of high importance. These areas correspond to the regions the model focused on during the decision-making process.

Color Mapping Interpretation:

- **Red and Yellow (Warm Colors):** Represent areas with the highest focus, indicating the parts of the image most relevant for classification.
- **Blue and Green (Cool Colors):** Represent regions of lower focus or importance.

The Grad-CAM heatmaps were generated and overlaid on the MRI images, showing areas of attention corresponding to different classes. These visualizations provide insight into how the model makes predictions and help confirm whether the model is focusing on relevant image features.

5.3.2 Grad-CAM Interpretation

The Grad-CAM heatmaps provided valuable insights:

- **Warm colors:** Highlighted brain areas that were likely tumors or regions affected by Alzheimer's disease.
- **Cool colors:** Showed regions the model considered less important for its prediction.
- **Success criteria**:
    - If the model focuses on relevant regions (e.g., tumor regions in brain MRI), it demonstrates a good decision-making process.
    - If irrelevant regions are highlighted, the model may need additional training to focus on the correct features.

This interpretability tool can guide us in understanding whether the model is focusing on biologically relevant regions and whether any adjustments are needed.

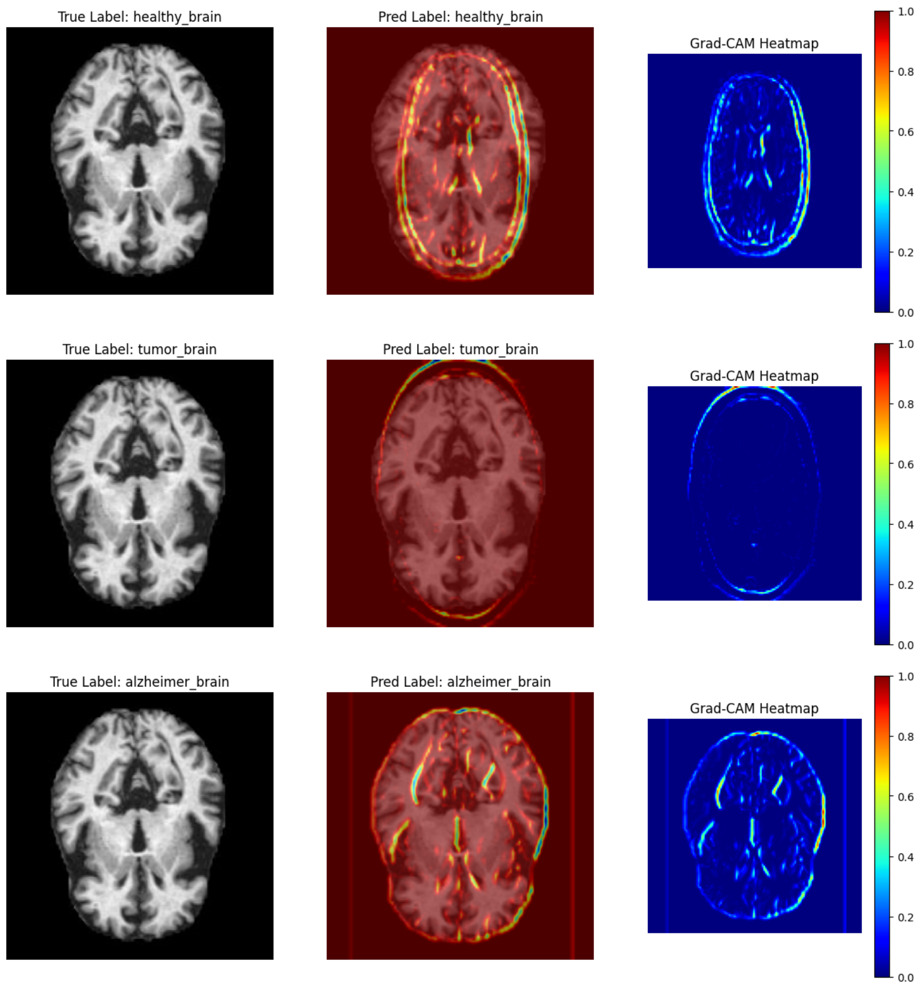


Figure 12: Grad-CAM overlay explanability images

5.4 Environmental Impact Evaluation

As part of responsible AI development, evaluating the environmental impact of training deep learning models is crucial. The computational requirements for training models like CNNs are enormous, leading to energy consumption and, consequently, carbon emissions.

5.4.1 Carbon Footprint Estimation

To estimate the environmental impact of training our model, we can use the **code carbon library**, which calculates the carbon footprint based on the energy consumption during training.

To use code carbon, you just have to put your script in between the initialization line **tracker.start()** and the closure line **tracker.stop()**

```
from codecarbon import EmissionsTracker

tracker = EmissionsTracker()
tracker.start()

# Training process here

tracker.stop()
```

Figure 13: CodeCarbon sample of implementation

The carbon footprint will depend on various factors such as hardware efficiency, energy source, and training duration. Since my laptop's resources were limited, I could not fully assess this part in the current environment, but it's an important aspect for future exploration. This step highlights the importance of considering sustainability in AI projects, especially when scaling up model training.

6 Conclusion

6.1 Model Performance

The model achieved an accuracy of 80.67% and a weighted F3 score of 80.60%, demonstrating reasonable performance for the given dataset. However, the model still struggles with distinguishing certain brain states (e.g., **healthy_brain vs. alzheimer_brain)**, suggesting that further model refinement is necessary, possibly through data augmentation, hyperparameter tuning, or more advanced architectures.

6.2 Explainability

The Grad-CAM visualizations helped improve our understanding of the model's decision-making process. The model appears to focus on relevant features, but further analysis and adjustments are needed to ensure that it is not overfitting to irrelevant regions as it is currently the case (the full images are set in red).

6.3 Environmental Impact

While the carbon impact calculation was not fully completed due to hardware limitations, it is an important area for future exploration. Responsible AI development should take into account both the performance of the model and its environmental footprint to get the **efficiency metric** (is further tuning of the model worth it?).

6.4 Future Work

- **Model Improvement:** Continue to refine the model, incorporating more advanced techniques and a larger dataset to improve classification accuracy.
- **Environmental Impact:** Use more powerful hardware for training and apply the codecarbon library to measure the true carbon impact.
- **Deployment:** Explore deploying the model in real-world healthcare settings, where it could assist medical professionals in diagnosing brain conditions.