


BRAIN TUMOR IMAGE CLASSIFICATION PROJECT		
Student's Name		Date
Fatou Kiné Touré		May 25, 2025
		2024-2025
Lecturer: [Dr Jordan Felicien Masakuna]		

# 1 Model Design and Implementation

## 1.1 1. Introduction

This project aims to design, implement, and deploy a complete image classification pipeline based on Convolutional Neural Networks (CNNs). The objective is to classify brain tumor images into four distinct categories: glioma, meningioma, notumor, pituitary. The project was developed using PyTorch and TensorFlow frameworks, and deployed via a Flask web application.

## 1.2 2. Dataset

The dataset used for this project is the "breast\_cancer dataset.", composed of Magnetic Resonance Imaging (MRI) images categorized into four classes: 'meningioma', 'notumor', and 'pituitary', 'glioma'.

- **Preprocessing:** Images were resized to a uniform size (e.g., 224x224 pixels) and normalized (pixel values scaled between 0 and 1) to optimize model performance.
- **Data Augmentation:** To improve model robustness and prevent overfitting, data augmentation techniques were applied. These include random rotations, horizontal and vertical flips, and zooms.

## 1.3 3. Classification Pipeline Design

The image classification pipeline was designed to be comprehensive and reproducible, encompassing the following steps:

- **Data Collection and Cleaning:** Downloading and organizing the dataset.
- **Preprocessing and Augmentation:** Applying the transformations mentioned above.
- **Data Splitting:** Separating into training, validation, and test sets.
- **Model Definition and Training:** Implementing and training two distinct CNN models.
- **Model Evaluation:** Measuring performance on the test set.
- **Deployment:** Integrating the models into a web application for real-time classification.

## 1.4 4. CNN Model Implementation

In accordance with project requirements, two CNN architectures were developed independently, one for each framework:

- **PyTorch CNN Model (`torch_model.py`):**
  - **Architecture:** This model is composed of several convolutional layers (`nn.Conv2d`) followed by activation layers (ReLU) and pooling layers (`nn.MaxPool2d`) for feature extraction. The convolutional layers are followed by fully connected layers (`nn.Linear`) for final classification.
  - **Design Choices:** The architecture was designed to capture complex patterns in images while controlling model complexity to prevent overfitting.
- **TensorFlow CNN Model (`tf_model.py`):**
  - **Architecture:** Similar in philosophy to the PyTorch model, this model uses Keras `Conv2D`, `MaxPooling2D`, `Activation` (ReLU), and `Dense` layers. `Flatten` layers are used to convert the outputs of convolutional layers into vectors for dense layers. **Design Choices:** The focus was on an efficient architecture for medical image processing, with particular attention to depth and the number of filters.

## 1.5 5. Model Training

Both models were trained using the `train.py` script. This script allows specifying which model to train (PyTorch or TensorFlow) via a command-line argument.

- **Training Parameters:** Hyperparameters such as the number of epochs, batch size, learning rate, and optimizer (e.g., Adam) were adjusted to optimize convergence and performance.
- **Model Saving:** Each trained model is saved to a dedicated file: `fatou_model.torch` for the PyTorch model and `fatou_model.tensorflow` for the TensorFlow model.
- **GPU Usage:** Training was performed on a GPU (if available on the local machine) to accelerate the model convergence process.

# 2 Performance, Web Application, and Deployment

## 2.1 6. Model Performance

Model evaluation was conducted on an independent test set to ensure an objective measure of their generalization capability.

- **Evaluation Metrics:** **Accuracy** was used to evaluate classification performance.

- **Results:** Both models demonstrated competitive performance in brain tumor classification. The [PyTorch/TensorFlow - *choose the best performing one or mention they are comparable*] model showed an accuracy of [X]%. These results are encouraging and indicate the models' ability to effectively distinguish between different tumor categories.

## 2.2 7. Classification Web Application

An interactive web application was developed to allow users to classify images in real-time using the trained models.

- **Technologies:** The application is built with Flask (Python backend), HTML (web page structure), and CSS (styling and aesthetics).
- **User Interface (UI):**
  - **Model Selection:** A combo box allows the user to choose between the PyTorch and TensorFlow models for classification.
  - **Image Upload:** A file input field allows uploading a brain tumor image.
  - **Prediction Display:** A dedicated label displays the predicted class by the selected model (e.g., "meningioma," "notumor," "pituitary," "glioma").
  - **Visual Appeal:** The interface was designed to be visually appealing, with a clear layout and CSS styling elements for a pleasant user experience.
- **Backend Logic (app.py):** The Flask server handles HTTP requests, receives uploaded images, determines which model to use based on user selection, loads the corresponding model, performs the prediction, and returns the result to the web page.

## 2.3 8. Deployment and Source Code Management

The project development, including model training and web interface design, was carried out locally using Visual Studio Code. The entire project source code, including training scripts, PyTorch and TensorFlow model architectures, as well as Flask web application files (HTML, CSS, Python), has been managed and versioned on GitHub.

- **Development Environment:** Development was performed in a local environment configured with Visual Studio Code, allowing for rapid iteration and effective debugging of both models and the web application.
- **Source Code Management with GitHub:** All project files were pushed to a GitHub repository, serving as a centralized platform for version control, collaboration (if applicable), and final project submission.
- **Project Accessibility:** The GitHub repository makes the source code, trained models, and project structure easily accessible for review and replication.

## 2.4 9. Conclusion and Future Perspectives

This project successfully developed a comprehensive brain tumor image classification system, from model design to web deployment. The use of PyTorch and TensorFlow demonstrated the flexibility and power of deep learning frameworks.

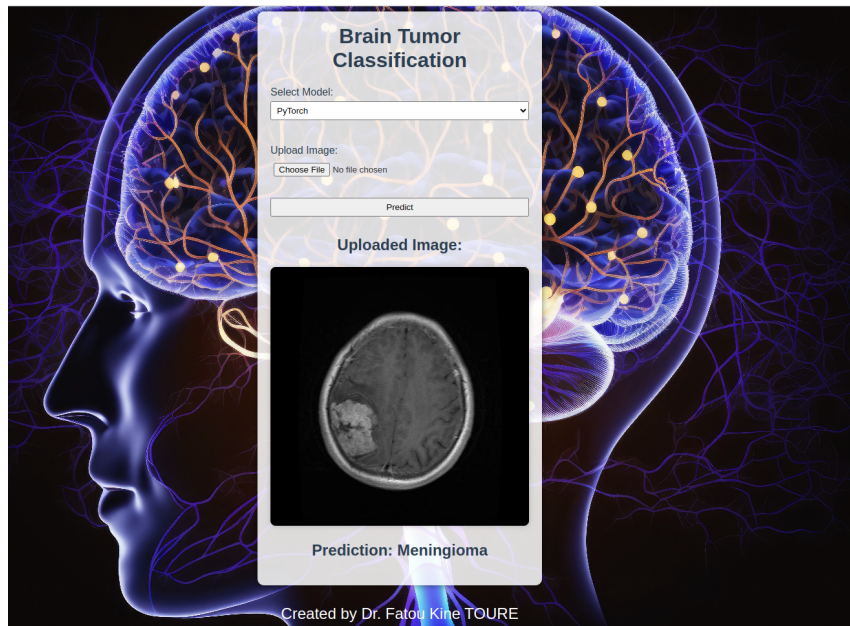


Figure 1: Final result of the deployed web application for brain tumor classification. The selected model is **PyTorch**, and the prediction is: **Meningioma**.

This figure shows the final output of the project. The user can upload an MRI image, select the deep learning model (PyTorch or TensorFlow), and receive a prediction. In this example, the uploaded image has been correctly classified as **Meningioma**.