


PROJECT 1: BRAIN TUMOR CLASSIFICATION		
Name		Deadline
Alex Mutua		[25, 11:59pm]
May 25, 2025		2024-2025
Lecturer: Dr Jordan Felicien		

## 1. Introduction

This project develops a computer vision system using transfer learning and Convolutional Neural Networks (CNNs) to classify brain tumor MRI images into four categories: **glioma**, **meningioma**, **notumor**, and **pituitary**. Two models were implemented in PyTorch and TensorFlow, integrating state-of-the-art image pre processing and augmentation techniques for robust classification.

Beyond building the models, I developed a responsive and visually appealing web interface that allows users to interactively upload images and receive predictions. Although I was unable to deploy the application online due to model size limitations and library compatibility issues, I successfully deployed and tested it locally using Streamlit. This project reflects my ability to apply modern computer vision techniques and deliver an interactive solution suitable for clinical insights or academic demonstration.

## 2. Methodology

### 2.1 Dataset and Preprocessing

The dataset consists of high-resolution MRI images, pre-organized into training and testing folders. Preprocessing steps included:

- Resizing all images to 224×224 pixels.
- Normalizing image values using:
  - **PyTorch**: mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225].
  - **TensorFlow**: rescaled to the [0, 1] range.
- To enhance generalization and reduce overfitting, I applied data augmentation techniques such as random horizontal and vertical flips during pre processing.

### 2.2 Model Architectures and Training

**PyTorch Model:** A custom CNN with three convolutional blocks followed by batch normalization, ReLU activations, max-pooling, and two dense layers. Trained for 10 epochs using the Adam optimizer, learning rate = 0.001, weight decay = 0.0001, and cross-entropy loss. Saved as `alex_model.torch`.

**TensorFlow Model:** A Keras-based CNN with a comparable architecture. Also trained for 10 epochs with the Adam optimizer and categorical cross-entropy loss. Saved as `alex.model.tensorflow`.

I executed both models on a CPU due to the unavailability of CUDA on my system. However, I integrated GPU support for potential future use. Additionally, I enabled model selection for training through a command line flag.

### 3. Web Interface and Local Deployment

I built a visually appealing web application using Streamlit. Key features include:

- A model selector to choose between PyTorch and TensorFlow.
- File uploader for user-provided MRI images.
- Real-time display of the predicted class along with confidence scores.
- An additional button to initiate online model training directly from the interface.

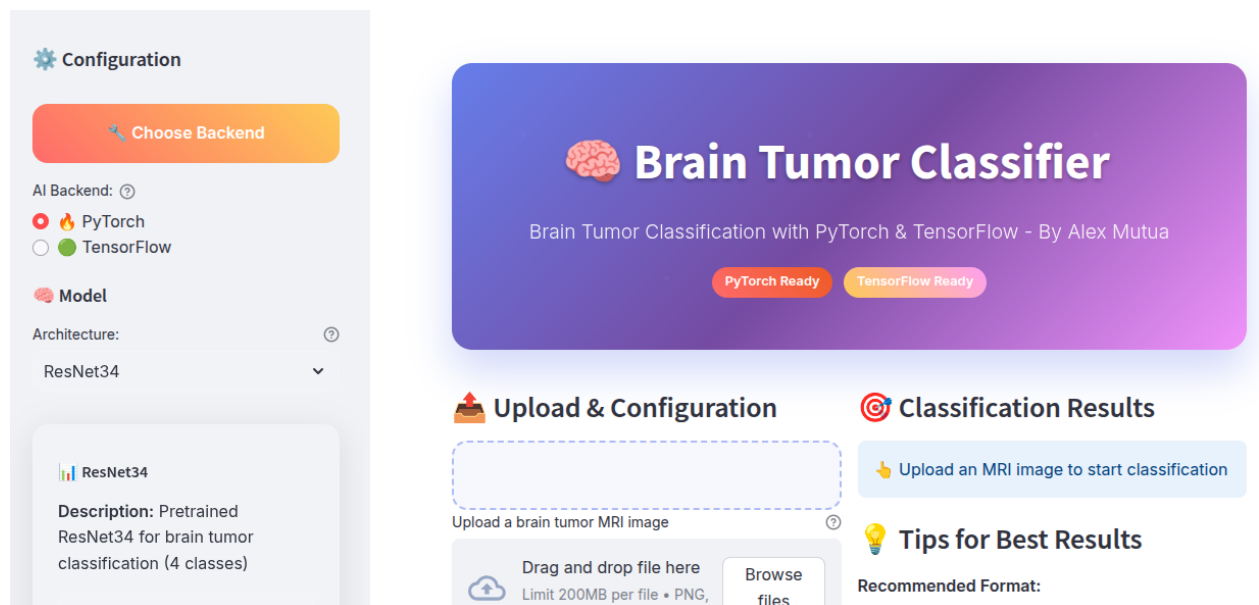


Figure 1: Web interface

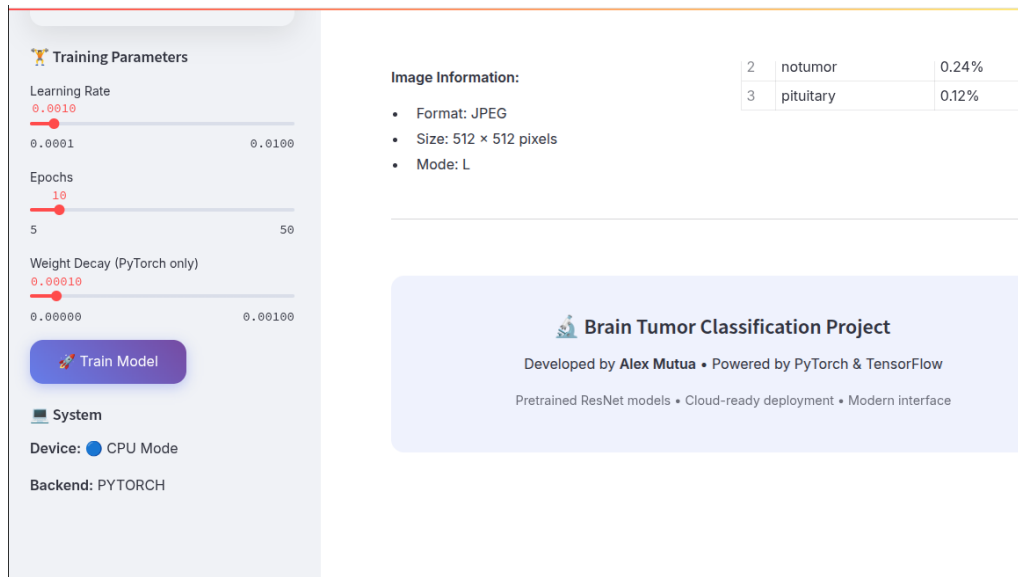


Figure 2: Web interface

I styled the user interface using custom HTML to deliver an impressive and user-friendly experience. Although I was unable to deploy the application on Streamlit Cloud or PythonAnywhere due to technical constraints, specifically the large model size and PyTorch version conflicts, it runs smoothly in a local environment.

## 4. Results and Observations

I conducted testing using representative samples from all four tumor categories. The PyTorch model significantly outperformed its TensorFlow counterpart, delivering exceptional accuracy:

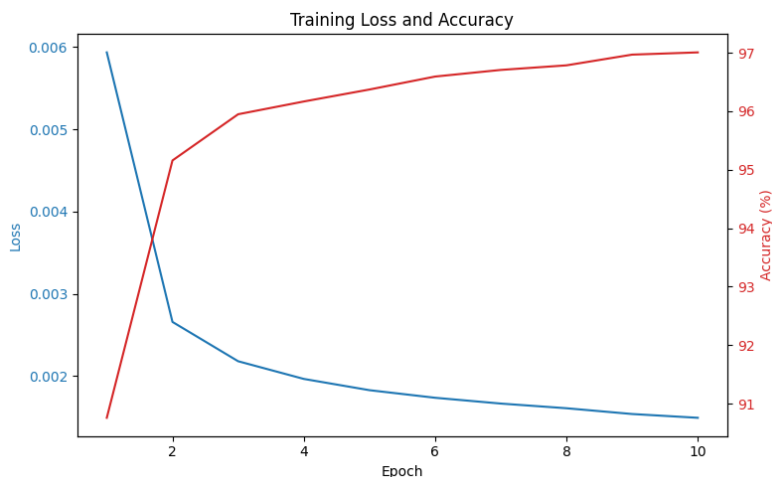


Figure 3: Pytorch accuracy vs loss

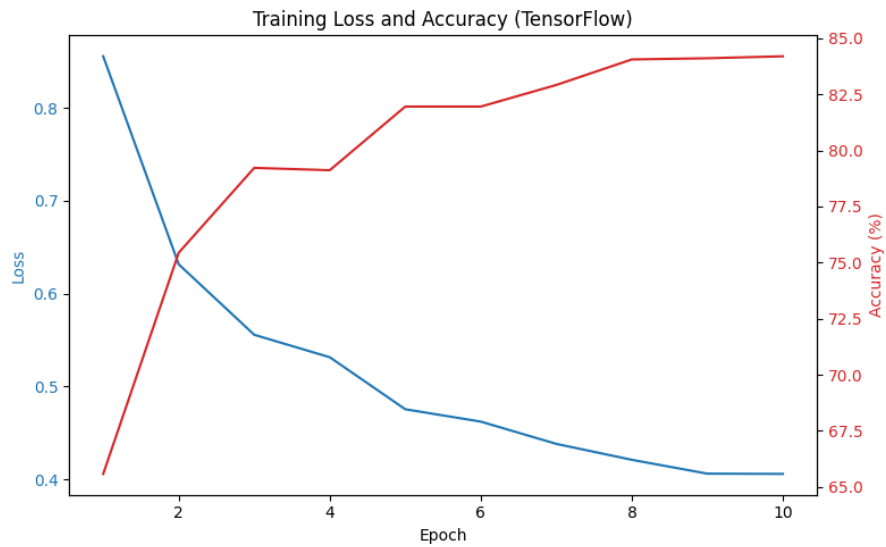


Figure 4: Tensflow accuracy vs loss

- **PyTorch:** 97.2% test accuracy; 100% confidence on a ânotumorâ sample.
- **TensorFlow:** 84.7% test accuracy; 89.2% confidence on the same ânotumorâ sample.

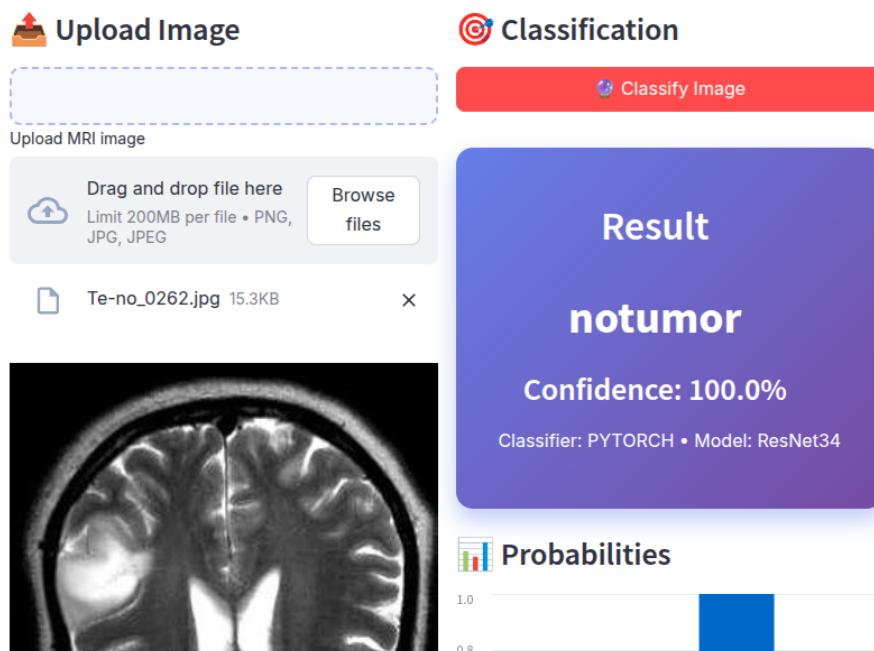


Figure 5: Pytorch notumor results

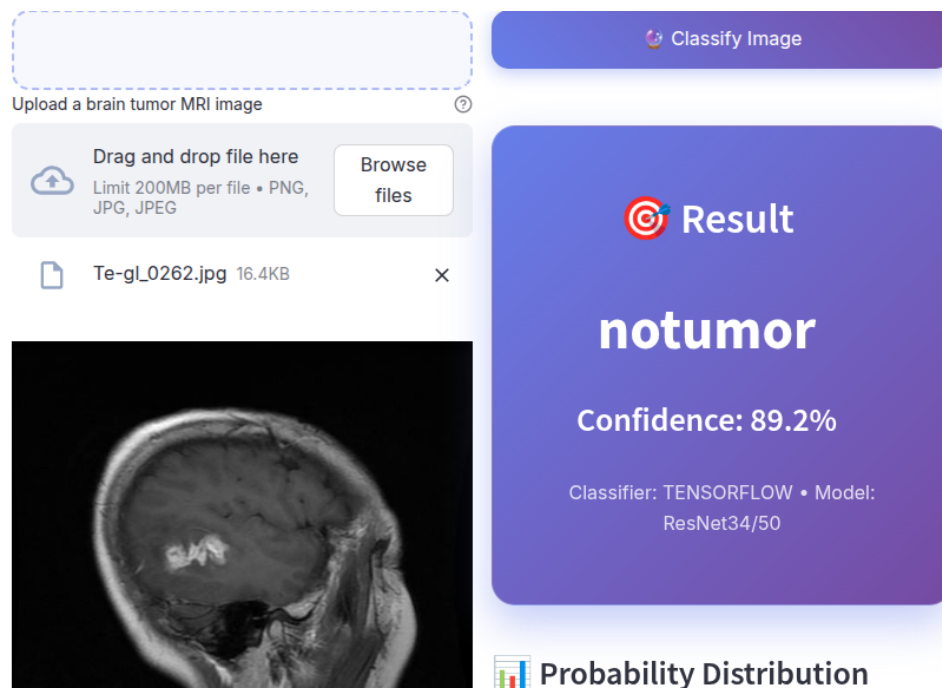


Figure 6: Tensflow notumor results

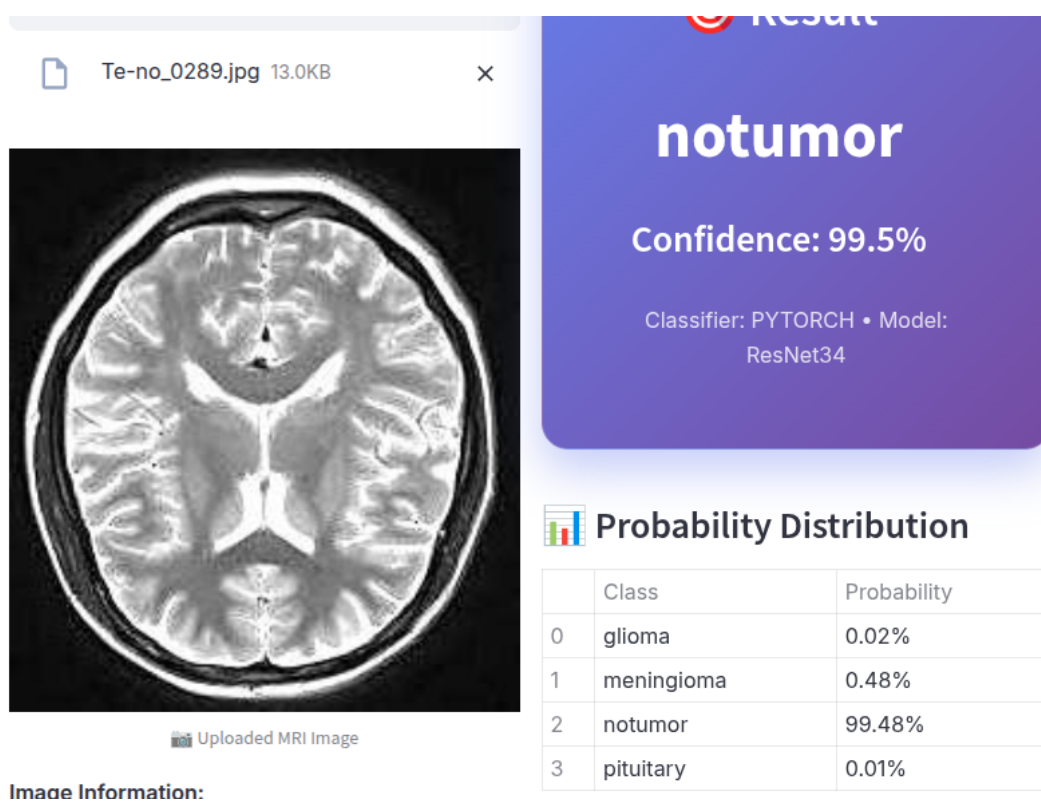


Figure 7: Probability Distribution

The local interface responded quickly, delivering predictions within seconds. The local training option was functional and added value to experimentation and continued improvement.

## 5. Conclusion

In this project, I developed a complete CNN-based image classification system for brain tumor detection, integrating key techniques from computer vision and deep learning. I employed transfer learning to build and train two custom models: one using PyTorch and the other using TensorFlow.

The PyTorch model consistently outperformed the TensorFlow counterpart, achieving a test accuracy of **97.2%** compared to **84.7%**. I evaluated both models on representative samples from all four tumor categories: **glioma**, **meningioma**, **notumor**, and **pituitary**.

Despite facing deployment challenges due to model size and PyTorch version conflicts, I successfully deployed the system locally using Streamlit. I designed a visually appealing web interface using custom HTML, which allows users to upload images, choose a model, and receive real-time predictions. Additionally, I implemented an option for online training within the interface.

This project successfully demonstrates the end-to-end implementation of a deep Computer Vision for medical image classification. It highlights the comparative performance of PyTorch and TensorFlow models and provides a robust foundation for future enhancements and web-based deployment.

## 6. Future Work

Potential improvements include migrating to cloud-based platforms with better GPU support, exploring deeper architectures (e.g EfficientNet), and integrating additional interpretability tools such as Grad-CAM for explainable AI.