ASSIGNMENT 1 ON COMPUTER VISION		
Student's Code		Deadline
MBUYI BIDIKUCHANCE Victor		**25/05/25**
May 25, 2025		2024-2025
		Lecturer: Jordan.F MASAKUNA

Brain Tumor Classification Project Report

Introduction

The Brain Tumor Classification project, developed for the AIMS Senegal Computer Vision course, employs deep learning to classify brain MRI scans into four categories: glioma, meningioma, notumor, and pituitary. This work aims to support medical diagnostics by automating tumor identification using pre-trained convolutional neural networks (CNNs) ResNet18 (PyTorch) and VGG16 (TensorFlow) integrated into a Flask web application.

1 Methodology

1.1 Dataset

The dataset, organized by organize_brain_tumor_dataset.py, contains 5712 training and 1311 testing MRI images across four classes, stored in /Users/mac/Desktop/CV A1/breast/projetFini/compilationFini/codetest/data/brain_tumor. Images are copied into training and testing subfolders, each with class-specific directories (glioma, meningioma, notumor, pituitary).

1.2 Models

- **PyTorch ResNet18 (pytorch_brain_tumor_cnn.py):** Uses a pre-trained ResNet18 with frozen convolutional layers. The final layer is replaced with a 4 class lin- ear layer, optimized with Adam (lr=0.001) and CrossEntropyLoss.
- **TensorFlow VGG16 (tensorflow_brain_tumor_cnn.py):** Employs a pre-trained VGG16 with frozen layers, followed by a classifier head (flatten, 256-unit dense ReLU, 0.5 dropout, 4 class softmax), optimized with Adam and categorical CrossEntropy.

1.3 Preprocessing

Images are resized to 224x224 pixels and processed with:

- **PyTorch:** Training includes random horizontal flips, 10-degree rotations, and normalization (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]). Testing uses only resizing and normalization.

- **TensorFlow:**Training applies horizontal flips, 10-degree rotations, and VGG16 preprocessing. Testing uses rescaling and VGG16 preprocessing.

  Batch size is 32 for both models.

1.4 Training

- **PyTorch:** Trained for 20 epochs, with loss monitored per epoch and test accuracy computed post-training.
- **TensorFlow:**Trained for 20 epochs (configurable), with validation accuracy tracked.

  Models are saved as Victor_model.torch and Victor_model.tensorflow.

1.5 Web Application

The Flask app (app.py) enables users to upload images, select a model, and view predictions and metrics. It preprocesses images using PyTorch transforms or TensorFlowâs VGG16 preprocessing, encoding images in base64 for display. Metrics are computed on the test set using scikit-learn.

2 Implementation and Results

Implemented in Python 3.8 on an Apple M1 (CPU), the project uses torch==2.1.2, torchvision==0.16.2, tensorflow==2.13.0, and flask==3.0.3 (see requirements.txt The Flask app handles HTTP requests and session management, rendering re- sults via index.html.

Model performance on the test set (1311 images) is summarized below:

Model	Accuracy (%)	Recall (%)	F1-Score (%)
PyTorch ResNet18	87	84.5	85.0
TensorFlow VGG19	92	91	89.5

Table 1: Model performance on test set.

2.1 Conclusion

This project successfully delivers an automated brain tumor classification system by integrating ResNet18 and VGG19 models into a Flask web application. The next major challenge will be deploying this solution to real-world platforms such as Heroku, Render, or Google Cloud Platform to ensure accessibility and scalability. Future improvements may also include fine-tuning additional layers, refining class balance.