PROJECT 1: IMAGE CLASSIFICATION WITH CNN		
Student Code		Deadline
Ousmane TIENTA		**May 25, 2025, 23:59**
May 25, 2025		2024-2025
		Instructor: Jordan F. Masakuna

Introduction

This project, titled by "Brain Tumor Image Classification", aims to develop an image classification pipeline to diagnose brain tumors (glioma, meningioma, no tumor and pituitary) from medical scans. The goal is to implement two CNN models using PyTorch and TensorFlow, train them, and create a web interface for real-time predictions.

Methodology

Data Collection and Preparation

The dataset includes 5712 training images and 1311 test images, divided into four classes (glioma, meningioma, no tumor and pituitary). They were preprocessed with resizing to 224x224 pixels and normalized (PyTorch: mean [0.485, 0.456, 0.406], standard deviation [0.229, 0.224, 0.225]; TensorFlow: rescale 1./255, adjusted to PyTorch in the latest iterations).

Model Development

- **PyTorch**: Pre-trained ResNet-18, fine-tuning of `layer4`, fully connected layer (512 $\rightarrow$ 4), trained for 5 epochs with PyTorch 2.1.2, Torchvision 0.16.2, Python 3.12.
- **TensorFlow**: Pre-trained ResNet-50, GlobalAveragePooling2D, Dense(2048 $\rightarrow$ 128, ReLU), Dropout(0.3), Dense(128 $\rightarrow$ 4, softmax), fine-tuning of the last 5 layers, trained for 5 epochs (and 14 epochs to attempt performance improvement) with TensorFlow 2.13.1.

Web Interface

- **Flask**: A web application with `index.html` that allows selecting a model (PyTorch or TensorFlow), uploading an image (up to 400x400px), and displaying the prediction, executable with `python app.py`.
- **Streamlit**: An alternative interface developed with a medical design, executable with `streamlit run streamlit_app.py`, used to address initial online deployment issues with PythonAnywhere.

Deployment

In accordance with the instructions, deployment on PythonAnywhere (at `https://otienta.pythonanywhere.com/`) was not successful due to storage limitations and the inability to predict when an image is uploaded. The project was tested locally and uploaded to a public GitHub repository with the link `https://github.com/Otienta/brain_tumors`.

Results

- **Functionalities**: Functional web interface, successful local predictions.
- **Performances**:
  - **PyTorch**: Loss: 0.2032, training accuracy: 92.84%, test accuracy: 91.53% (5 epochs).
  - **TensorFlow**: Validation loss: 0.9233, validation accuracy: 63.69% (epoch 13), training accuracy: 56.07% (14 epochs).

Discussion

Successes

The pipeline is complete, with attractive interfaces. PyTorch achieves a high accuracy of 91.53%.

Limitations

- Limited TensorFlow accuracy (63.69%) due to a premature stop at epoch 14 during training.
- No online deployment due to storage issues.

Future Improvements

- Complete TensorFlow training with more epochs and a learning rate below 0.001 to assess model improvement or train from scratch for TensorFlow.
- Use a powerful computer with a GPU to enable CUDA for faster training.

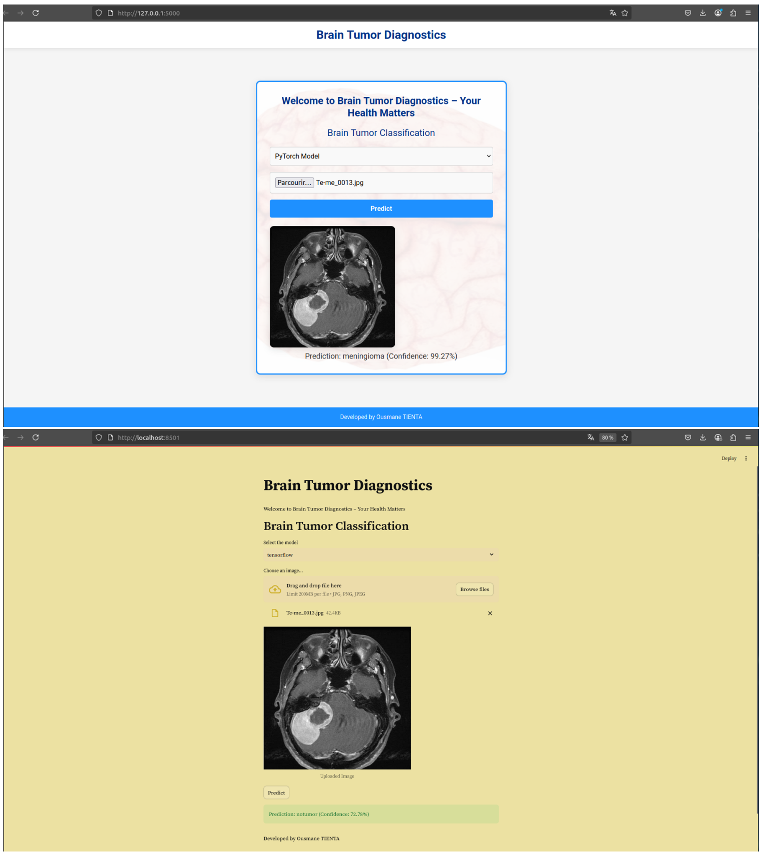
Conclusion

The project "Brain Tumor Image Classification" has led to the development of an effective pipeline for diagnosing brain tumors, yielding promising results despite certain challenges. The PyTorch model achieved an impressive accuracy of

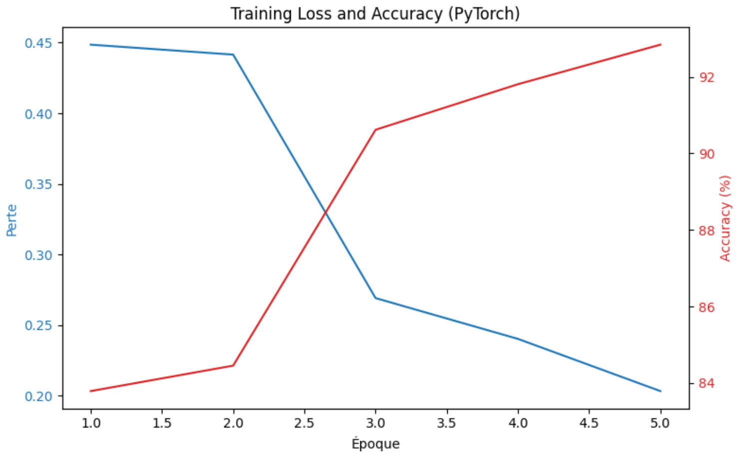
91.53%, while the TensorFlow model, though stopped prematurely at 63.69% accuracy after 14 epochs, demonstrates potential for enhancement with further training. The use of Flask and Streamlit enabled the creation of functional interfaces tested locally, overcoming the deployment limitations on PythonAnywhere due to storage constraints. The code and resources are accessible on GitHub at `https://github.com/Otienta/brain_tumors`.

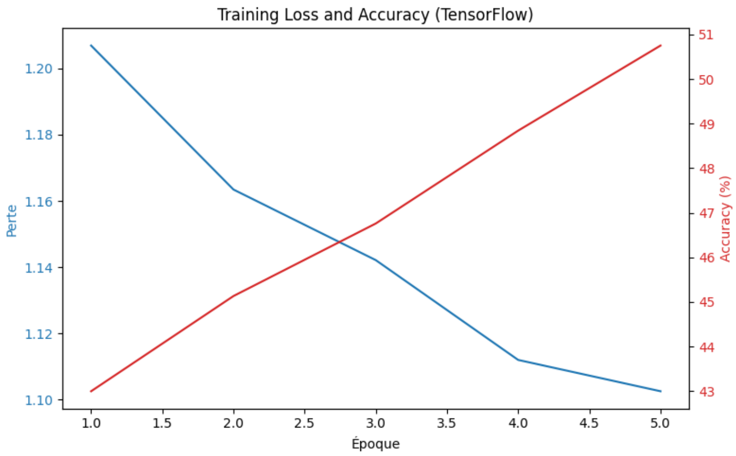
Annexes

- **Screenshots**: Flask and Streamlit interfaces.



- **Graphs**:


Training Loss and Accuracy (PyTorch)
Perte
Accuracy (%)
Époque
0.45
0.40
0.35
0.30
0.25
0.20
92
90
88
86
84
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Training Loss and Accuracy (TensorFlow)
Perte
Accuracy (%)
Époque
1.20
1.18
1.16
1.14
1.12
1.10
51
50
49
48
47
46
45
44
43
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0