


ASSIGNMENT [1] ON [COMPUTER VISION]		
Student's Code		Deadline
[Anna NIANE]		25th May 2025 at 11.59pm]
May 25, 2025		2024-2025
Lecturer: [Jordan Felicien Masakuna]		

1 Introduction

In this project we develop a deep learning system for classifying brain tumors from MRI images into four categories: glioma, meningioma, notumor, and pituitary. The main objectives are to achieve high classification accuracy using convolutional neural networks (CNNs) implemented in PyTorch and TensorFlow, optimize training efficiency, and deploy a Flask web application for real-time predictions. The project is controlled on GitHub (https://github.com/AnnaNIANE/project_breast_cancert) and you can see the interface of the flask app at (https://annaniane.github.io/project_breast_cancert/) .

2 Dataset

The project utilizes the breast cancer Dataset comprising approximately 5,716 training images and 1,315 testing images across four classes: glioma, meningioma, notumor, and pituitary. Images are resized and normalized for model input.

3 Methodology

Two CNN models were developed to perform the classification task, with architectures designed to process $224 \times 224 \times 3$ RGB images:

- **PyTorch CNN** (`models/cnn_torch.py`): A custom `CNN_Torch` class inheriting from `torch.nn.Module`, with three convolutional layers (16, 32, and 64 filters, each with 3×3 kernels and ReLU activation), followed by max-pooling layers (2×2). The feature maps are flattened to $64 \times 28 \times 28$ (after three pooling operations reducing spatial dimensions from 224 to 28), followed by a fully connected layer (128 units, ReLU), a dropout layer (0.5), and an output layer (4 units). The model is trained using the Adam optimizer with a learning rate of 0.001, weight decay of 0.0001, and CrossEntropyLoss.
- **TensorFlow CNN** (`models/cnn_tensorflow.py`): A sequential Keras model with a similar architecture: three convolutional layers (16, 32, and 64 filters, 3×3 kernels, ReLU, same padding), max-pooling (2×2), a flatten layer, a dense layer (128 units, ReLU), a dropout layer (0.5), and an output layer (4 units, softmax). The model is compiled with the Adam optimizer (learning rate 0.001) and sparse categorical crossentropy loss.

The training is managed by `models/train.py`, which supports both frameworks through `TrainerTorch` and `TrainerTensorFlow` classes. The training process runs for 10 epochs (we can change it via command-line arguments) with a batch size defined in `utils/preprocessing.py`. The PyTorch trainer saves the model as `anna_model.torch`. The TensorFlow trainer saves the model as `anna_model.tensorflow`.

4 Results

The models were evaluated on the test set, achieving the following performance metrics:

Framework	Test Accuracy (%)	Test Loss
PyTorch	85.66%	0.3440
TensorFlow	87.72%	0.3182

Note:

Evaluation metrics are logged in `pytorch_evaluation.txt` and `tensorflow_evaluation.txt`. The training history (accuracy and loss) plotted and saved as `pytorch_training_history.png` and `tensorflow_training_history.png`.

5 4. Possible Improvements

- Integration of **Data Augmentation**
- Use of **Transfer Learning** (ResNet, MobileNet) for better accuracy since it is in the sensitive domain of health.
- Application of **stratified split** to balance the classes

6 Conclusion

In this project we try to build a clean and modular architecture for an image classification task from scratch ,the models classifies brain tumors with accuracies of 85.66% (PyTorch) and 87.72% (TensorFlow). We creat a Flask web app for user interaction. Future improvements include enhancing accuracies through advanced data augmentations and pre trained model (like ResNet).