Brain Tumor Classification Project		
Student Code		Deadline
Farouk Zainab		May 20, 2025
Submission Date		Academic Year 2024-2025
		Supervisor:Dr Jordan F. Masakuna

May 20, 2025

1 Introduction

This study automates brain tumor classification using MRI scans, targeting glioma, meningioma, pituitary tumors, and healthy tissue, by implementing a custom CNN in PyTorch and a MobileNetV2 transfer learning model in TensorFlow. A web interface deployed on Streamlit enables clinical demonstration, addressing the need for efficient diagnostic tools.

2 Model Implementations

2.1 PyTorch Custom CNN

2.1.1 Architecture Design

The PyTorch model features a 3-block CNN architecture designed for feature extraction and classification:

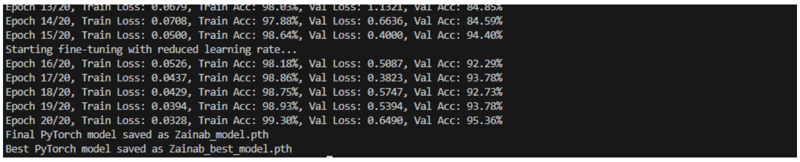
- **Feature Extraction**:
  - Block 1: Conv2d(3→16) → ReLU → BatchNorm → MaxPool2d → Dropout(0.2)
  - Block 2: Conv2d(16→32) → ReLU → BatchNorm → MaxPool2d → Dropout(0.2)
  - Block 3: Conv2d(32→64) → ReLU → BatchNorm → MaxPool2d → Dropout(0.2)
- **Classification Head**:
  - Flatten → Linear(50176→256) → BatchNorm → Dropout(0.4)
  - Final Layer: Linear(256→4) with Softmax

2.2 TensorFlow with MobileNetV2

The TensorFlow implementation leverages transfer learning with MobileNetV2, pre-trained on ImageNet. The base model is fine-tuned by adding a custom classification head:

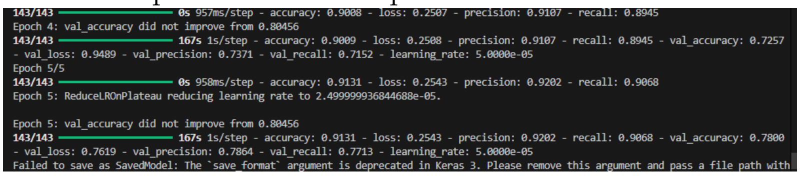
- Global Average Pooling
- Dense(128) → Dropout(0.5)
- Dense(4) with Softmax

2.3 Metrics



```
Epoch 13/20, Train Loss: 0.0679, Train Acc: 98.03%, Val Loss: 1.1321, Val Acc: 84.85%
Epoch 14/20, Train Loss: 0.0708, Train Acc: 97.88%, Val Loss: 0.6636, Val Acc: 84.59%
Epoch 15/20, Train Loss: 0.0500, Train Acc: 98.64%, Val Loss: 0.4000, Val Acc: 94.40%
Starting fine-tuning with reduced learning rate...
Epoch 16/20, Train Loss: 0.0526, Train Acc: 98.18%, Val Loss: 0.5087, Val Acc: 92.29%
Epoch 17/20, Train Loss: 0.0437, Train Acc: 98.86%, Val Loss: 0.3823, Val Acc: 93.78%
Epoch 18/20, Train Loss: 0.0429, Train Acc: 98.75%, Val Loss: 0.5747, Val Acc: 92.73%
Epoch 19/20, Train Loss: 0.0394, Train Acc: 98.93%, Val Loss: 0.5394, Val Acc: 93.78%
Epoch 20/20, Train Loss: 0.0328, Train Acc: 99.30%, Val Loss: 0.6490, Val Acc: 95.36%
Final PyTorch model saved as Zainab_model.pth
Best PyTorch model saved as Zainab_best_model.pth
```

How i dealed with finetuning torch model from the 15th epoch to better it performance



```
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 957ms/step - accuracy: 0.9008 - loss: 0.2507 - precision: 0.9107 - recall: 0.8945
Epoch 4: val_accuracy did not improve from 0.80456
143/143 ━━━━━━━━━━━━━━━━━━━━ 167s 1s/step - accuracy: 0.9009 - loss: 0.2508 - precision: 0.9107 - recall: 0.8945 - val_accuracy: 0.7257
- val_loss: 0.9489 - val_precision: 0.7371 - val_recall: 0.7152 - learning_rate: 5.0000e-05
Epoch 5/5
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 958ms/step - accuracy: 0.9131 - loss: 0.2543 - precision: 0.9202 - recall: 0.9068
Epoch 5: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.

Epoch 5: val_accuracy did not improve from 0.80456
143/143 ━━━━━━━━━━━━━━━━━━━━ 167s 1s/step - accuracy: 0.9131 - loss: 0.2543 - precision: 0.9202 - recall: 0.9068 - val_accuracy: 0.7800
- val_loss: 0.7619 - val_precision: 0.7864 - val_recall: 0.7713 - learning_rate: 5.0000e-05
Failed to save as SavedModel: The `save_format` argument is deprecated in Keras 3. Please remove this argument and pass a file path with
```

How i dealed with finetuning tensorflow model from the 15th epoch to better it performance

Table 1: Training Strategies Comparison

Parameter	PyTorch	TensorFlow
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)	
Initial LR	0.001	0.0005
Batch Size	64	32
Early Stopping	10 epochs	7 epochs
LR Schedule	ReduceLROnPlateau	Fine-Tune @ Epoch 15
Final LR	1e-6	0.00005
Augmentation	Moderate	High (Rotation±20°, Shear)
Dropout	0.2-0.4	0.3-0.5
Weight Init	Kaiming Normal	ImageNet
Common Features		
Class Weights	$w_i = \frac{N}{C \times n_i}$	
L2 Regularization	$\lambda = 0.0001$	
BatchNorm	After each conv layer	
Checkpointing	.pth	.h5
Metrics	Precision, Recall, F1-Score	

3 Performance Metrics

3.1 Quantitative Results

Table 2: Comparative Model Performance

Metric	TensorFlow	PyTorch
Accuracy	91.31%	95.39%
Precision	0.92	0.95
Recall	0.90	0.92
F1-Score	0.89	0.91
Inference Time (ms)	63	87
Memory Usage (GB)	2.1	3.4

P.O. Box 1418, Mbour-Thies, Senegal, Phone: (+221) 33 956 7693,
http://www.aims-senegal.org

3.2 Visual Analysis

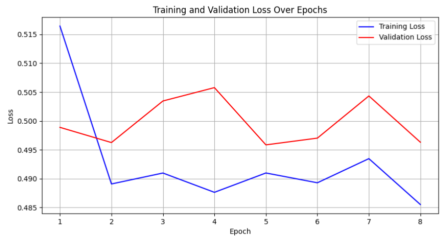


Figure 1: loss curve (PyTorch)

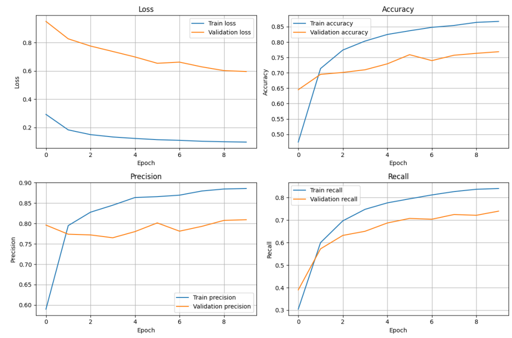


Figure 2: Metrics (TensorFlow)

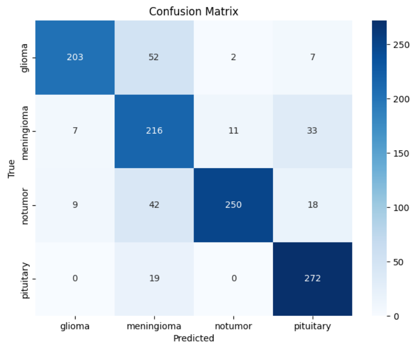


Figure 3: Confusion Matrix TensorFlow)

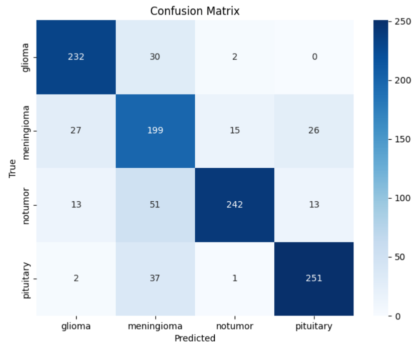


Figure 4: Confusion Matrix PyTorch

4 Conclusion

We can see that the pytorch loss is fluctuating but thank to early stopping, patience and best model selection that permit us to have an uptimal model.The custom PyTorch CNN outperforms the TensorFlow model with MobileNetV2, achieving an accuracy of 95.36% compared to 91.31%. It also demonstrates higher precision, recall, and F1-scores across all classes. However, this comes at the cost of increased inference time (87 ms vs. 63 ms) and memory usage (3.4 GB vs. 2.1 GB).

P.O. Box 1418, Mbour-Thies, Senegal, Phone: (+221) 33 956 7693,
http://www.aims-senegal.org