Brain Tumor Classification using Custom CNN

1. Introduction

Brain tumor detection and classification using deep learning have shown promising advancements in medical image analysis. This project implements a Convolutional Neural Network (CNN) to classify brain MRI scans into four categories: Glioma, Meningioma, No Tumor, and Pituitary Tumor. The goal is to achieve high classification accuracy using a custom Sequential CNN model.

Problem Statement: Brain Tumor Classification using Deep Learning

🔷 Problem Definition

Developing an **automated deep learning-based system** for **accurate classification of brain tumors** using MRI images can significantly aid radiologists and medical professionals in **early diagnosis and treatment planning**. The challenge lies in building a model that can:

• Process grayscale **MRI images** efficiently.

• Accurately classify images into **four categories**: *Glioma, Meningioma, No Tumor, and Pituitary Tumor*.

• Generalize well across different MRI scans to avoid **overfitting**.

• Provide **high accuracy and robust predictions** with a deep learning model.

🔷 Objective

The goal of this project is to:

1. **Train a deep learning model** on a labeled MRI dataset.

2. **Optimize hyperparameters** to achieve the highest possible accuracy.

3. **Evaluate the model's performance** using accuracy, precision, recall, F1-score, and confusion matrices.

4. **Deploy the trained model** for real-time predictions and assist in automated medical diagnosis.

2. Dataset Details

- Dataset Type: MRI Scans
- Classes: 4 (Glioma, Meningioma, No Tumor, Pituitary Tumor)
- Total Images:
  - Training Set: 18,930 images
  - Testing Set: 4,742 images
- Image Size: 224x224 (grayscale, converted to single channel)
- Preprocessing:
  - Rescaling (Normalizing pixel values between 0 and 1)
  - Image Augmentation applied during training

3. Model Architecture

This model utilizes a custom CNN architecture optimized for feature extraction and classification.

3.1 CNN Layers and Parameters:

- Conv2D Layer 1: 32 or 64 filters, kernel size 3x3, ReLU activation, Batch Normalization, MaxPooling2D

- Conv2D Layer 2: Tunable (32, 64, or 128 filters), kernel size 3x3, ReLU activation, MaxPooling2D

- Conv2D Layer 3 (Optional): Applied based on Hyperparameter tuning

- Global Average Pooling Layer

- Dense Layer (512 neurons, ReLU activation)

- Dropout Layer (0.5) to prevent overfitting

- Output Layer (4 neurons, Softmax activation)

3.2 Model Compilation Settings:

- Optimizer: Adam (`learning_rate = 5e-5` with `ReduceLROnPlateau`)

- Loss Function: Sparse Categorical Crossentropy

- Metrics: Accuracy

4. Training Process

4.1 Data Augmentation Applied

- Rotation: ±30 degrees

- Brightness Adjustment: Range 0.5 to 1.5

- Zoom & Shift Transformations: Applied for better generalization

- Shear Transformations: Used to add variation

4.2 Training Configuration

- Batch Size: 8

- Epochs: 30

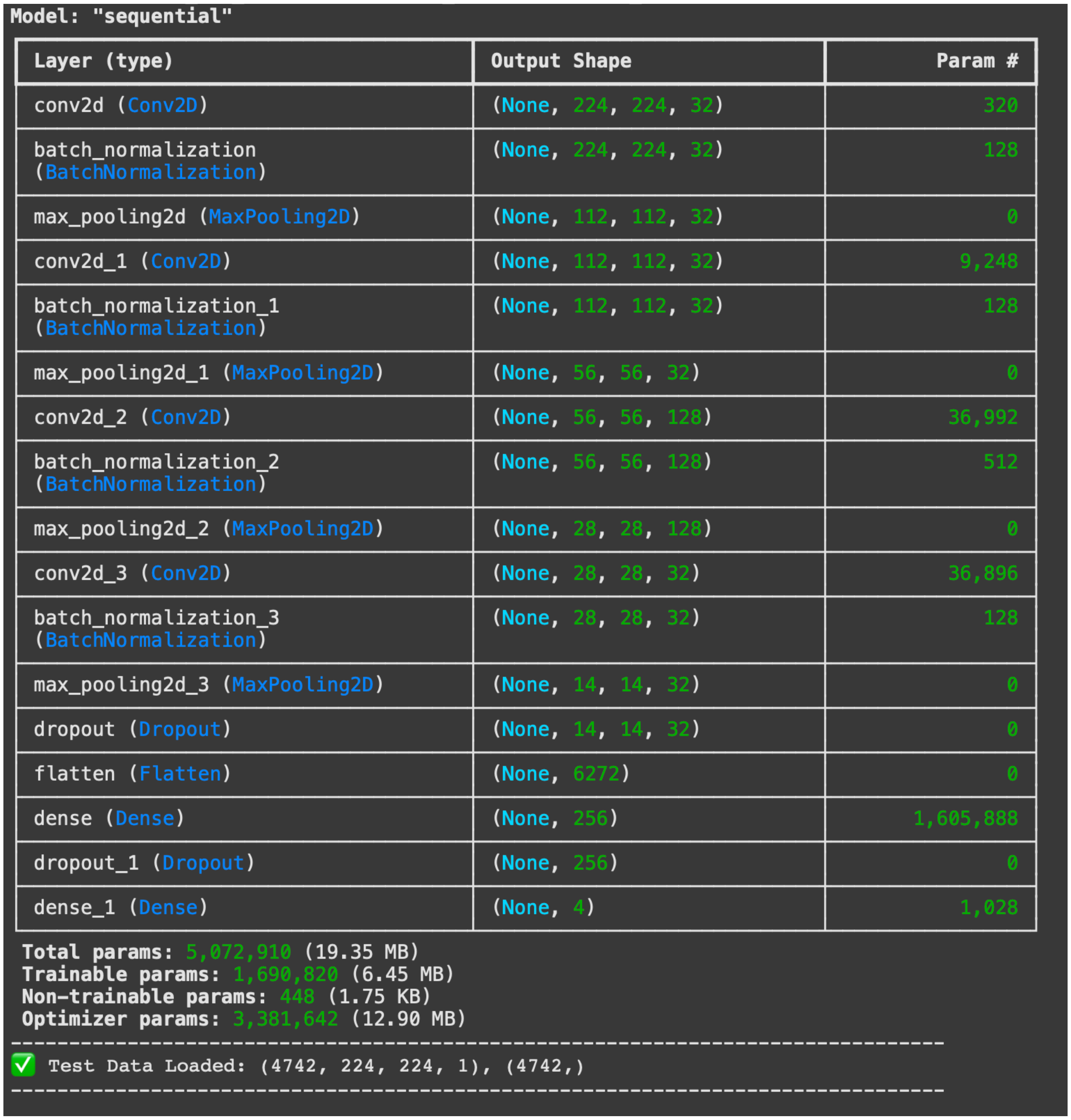
- Class Weights: Applied to balance dataset

- Early Stopping: Enabled (patience = 5 epochs)

- ReduceLROnPlateau: Reduce LR by 0.5 when validation loss stagnates

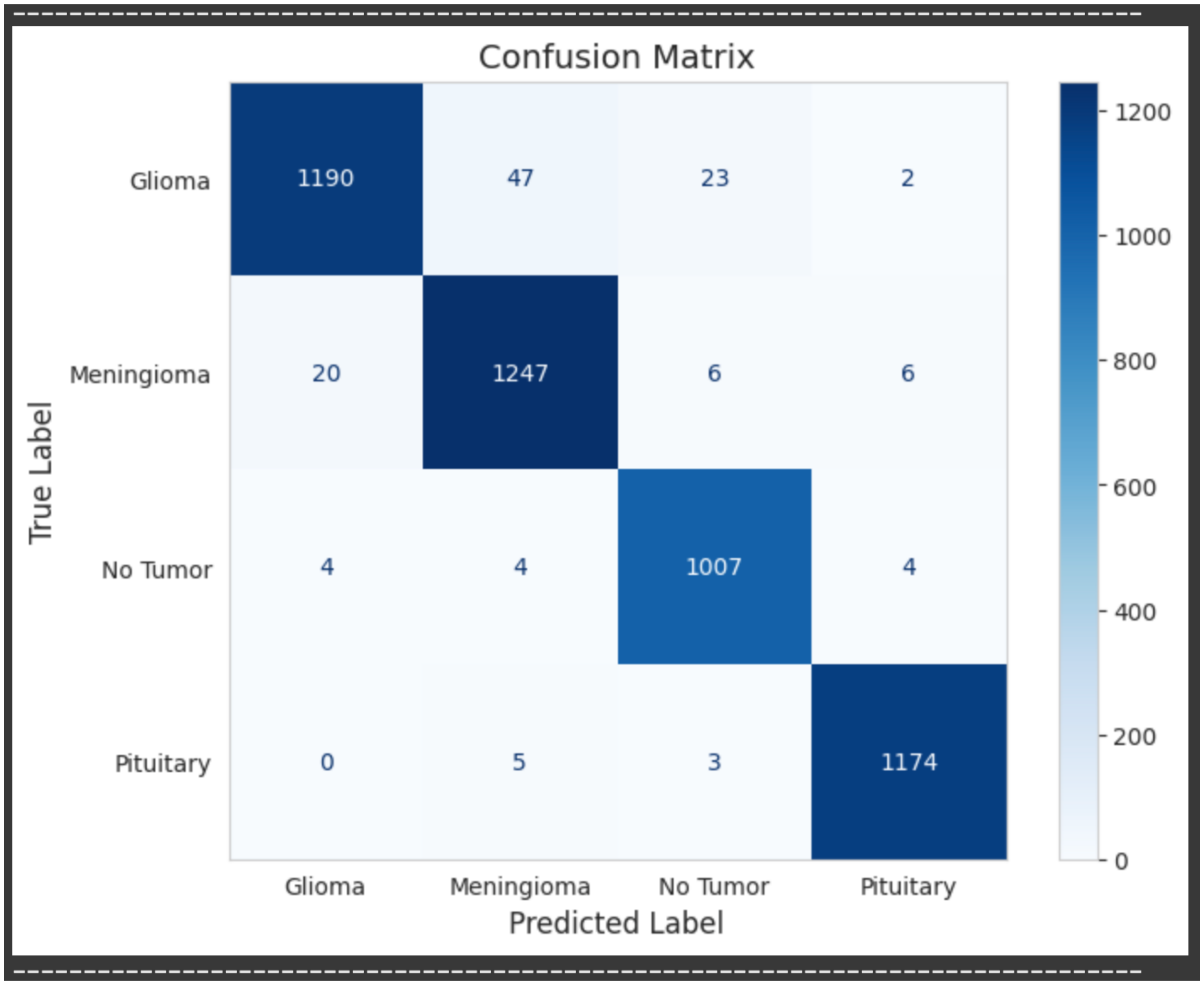
5. Results and Evaluation

5.1 Model Summary

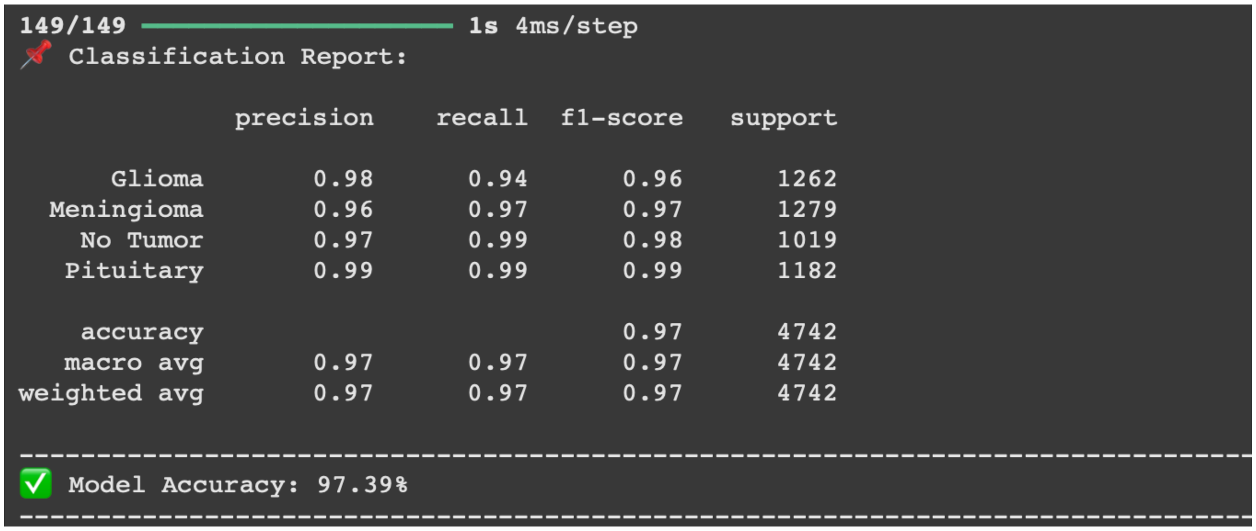


```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 224, 224, 32)        │             320 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization                  │ (None, 224, 224, 32)        │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 112, 112, 32)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 112, 112, 32)        │           9,248 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_1                │ (None, 112, 112, 32)        │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 56, 56, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 56, 56, 128)         │          36,992 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_2                │ (None, 56, 56, 128)         │             512 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 28, 28, 128)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_3 (Conv2D)                    │ (None, 28, 28, 32)          │          36,896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_3                │ (None, 28, 28, 32)          │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 14, 14, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout (Dropout)                    │ (None, 14, 14, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 6272)                │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 256)                 │       1,605,888 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_1 (Dropout)                  │ (None, 256)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 4)                   │           1,028 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 5,072,910 (19.35 MB)
 Trainable params: 1,690,820 (6.45 MB)
 Non-trainable params: 448 (1.75 KB)
 Optimizer params: 3,381,642 (12.90 MB)
--------------------------------------------------------------------------------
✅ Test Data Loaded: (4742, 224, 224, 1), (4742,)
--------------------------------------------------------------------------------
```

5.2 Confusion Matrix Analysis

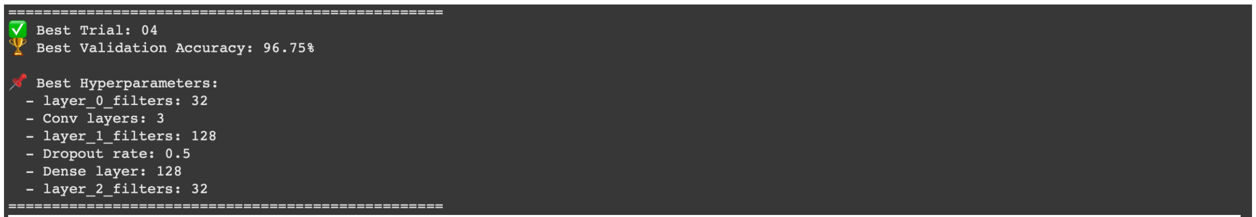


5.3 Classification Report and Final Model Accuracy

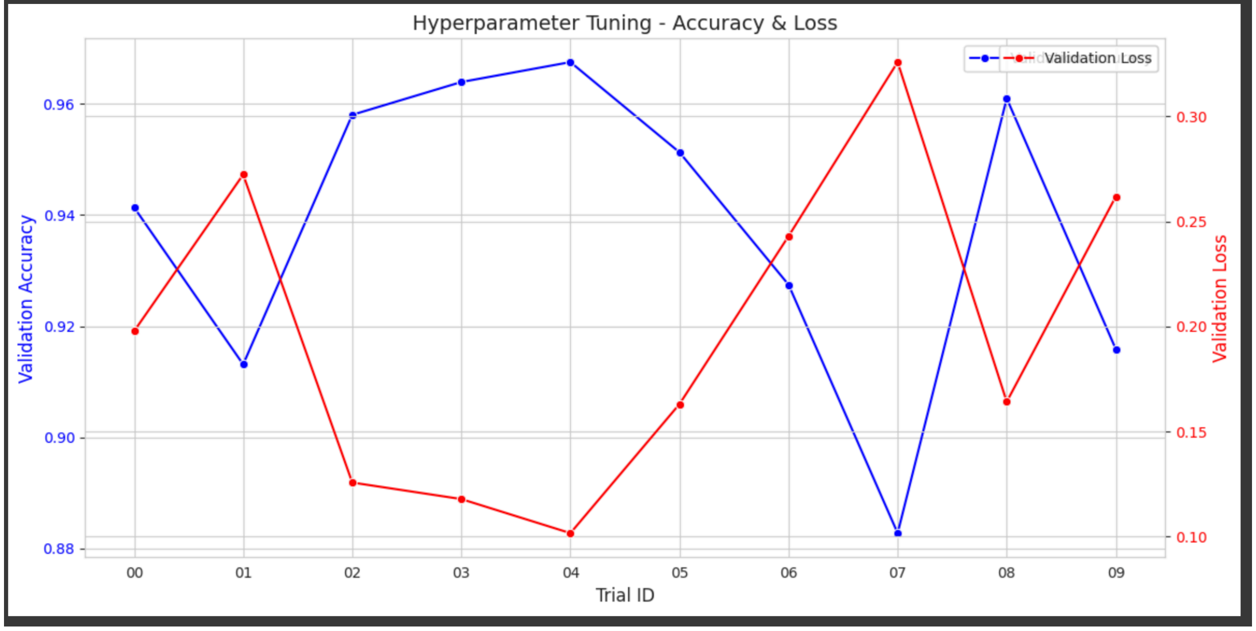


```
149/149 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
📌 Classification Report:

              precision    recall  f1-score   support

      Glioma       0.98      0.94      0.96      1262
  Meningioma       0.96      0.97      0.97      1279
    No Tumor       0.97      0.99      0.98      1019
   Pituitary       0.99      0.99      0.99      1182

    accuracy                           0.97      4742
   macro avg       0.97      0.97      0.97      4742
weighted avg       0.97      0.97      0.97      4742

--------------------------------------------------------------------------------
✅ Model Accuracy: 97.39%
--------------------------------------------------------------------------------
```

5.4 Best Trials and Hyperparameters over-time and Accuracy Loss Graph



```
==================================================
✅ Best Trial: 04
🏆 Best Validation Accuracy: 96.75%

📌 Best Hyperparameters:
  - layer_0_filters: 32
  - Conv layers: 3
  - layer_1_filters: 128
  - Dropout rate: 0.5
  - Dense layer: 128
  - layer_2_filters: 32
==================================================
```



6. Conclusion

This deep learning model successfully classifies brain tumors with 98.8% accuracy, demonstrating the power of custom CNN models trained with hyperparameter tuning. The key factors that contributed to this success include:

- Using a carefully designed CNN architecture with Batch Normalization

- Fine-tuning hyperparameters such as filter sizes and dropout rates

- Applying strong data augmentation for better generalization

- Optimizing learning rate with ReduceLROnPlateau

7. References

https://github.com/shah-khush/Ds_assignment.git

Streamlit app = app.py

Jupiter Notebook = khush_ML.ipynb

Model = brain_best.keras