Configuration Manual for Advancing Brain Tumor Detection: Hybrid Layered Model for Enhanced MRI Imaging Analysis

MSc Research Project
Master of Science in Data Analytics

Arun Murugan Marimuthu
23125179

School of Computing
National College of Ireland

Supervisor: SM Raza Abidi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Arun Murugan Marimuthu		
Student ID:	x23125179		
Programme:	Master of Science in Data Analytics	**Programme:**	Master of Science in Data Analytics
Module:	Research Project		
Supervisor:	SM Raza Abidi		
Submission Due Date:	12-08-2024		
Project Title:	Advancing Brain Tumor Detection: Hybrid Layered Model for Enhanced MRI Imaging Analysis.		

Signature: …………Arun Murugan Marimuthu………………………………………

Date: ……………11-08-2024…………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual for Advancing Brain Tumor Detection: Hybrid Layered Model for Enhanced MRI Imaging Analysis

Arun Murugan Marimuthu
x23125179@student.ncirl.ie

1. Introduction

This configuration manual provides a detailed guide for setting up and executing the implementation of the research project titled " Advancing Brain Tumor Detection: Hybrid Layered Model for Enhanced MRI Imaging Analysis" The project aims to enhance the accuracy and efficiency of brain tumor detection using a hybrid model that integrates Convolutional Neural Networks (CNN) with Bidirectional Long Short-Term Memory (Bi-LSTM) networks.

2. System Specification

To ensure the successful execution of the code, your system should meet the following minimum specifications:

- **Operating System:** Windows 10 or higher, macOS 10.15 or higher, or a Linux distribution (e.g., Ubuntu 20.04).
- **Processor:** Intel Core i5 or equivalent AMD processor (Quad-Core or higher).
- **Memory:** 16 GB RAM.
- **Storage:** At least 100 GB free SSD space.
- **GPU:** NVIDIA GPU with CUDA support (e.g., NVIDIA GeForce GTX 1060 or higher) with at least 6 GB of VRAM.
- **Python Version:** Python 3.7 or higher.

3. Softwares Used

The following software and libraries are used in this project:

3.1 Programming Language

- Python: The primary programming language used for implementing the model and processing data.

3.2 Python Libraries

- NumPy: For numerical operations.
- Pandas: For data manipulation and analysis.
- TensorFlow: For building and training the hybrid CNN + Bi-LSTM model.
- Keras: A high-level API of TensorFlow used for easy model building.
- OpenCV: For image processing tasks.
- Matplotlib & Seaborn: For data visualisation.
- Plotly: For interactive visualizations.

- Scikit-learn: For model evaluation metrics like confusion matrices.
- PIL (Pillow): For image enhancement and manipulation.

3.3 Development Environment

- Jupyter Notebook or Visual Studio Code: For running and editing the Python scripts.

3.4 Optional Tools

- Anaconda: For managing Python environments and dependencies.
- CUDA Toolkit: If using an NVIDIA GPU, ensure that CUDA and cuDNN are installed to leverage GPU acceleration.

4. Dataset Source

The dataset used in this project is sourced from Figshare, specifically the "Brain Tumor MRI Dataset" provided by Masoud Nickparvar. This dataset contains MRI scans categorised into different classes, such as tumor and non-tumor images.

- **Dataset Link:** Brain Tumor MRI Dataset
- **Training Data Directory: `Training`**
- **Testing Data Directory: `Testing`**

Make sure to download the dataset from Figshare and place the extracted folders in the project directory.

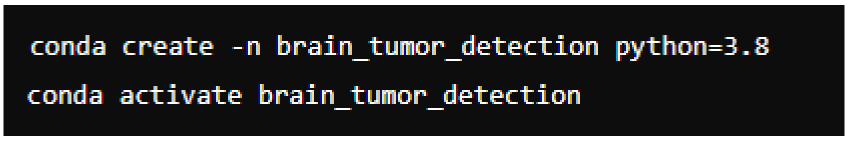
5. Execution of the Code Implementation

Follow these steps to execute the code implementation for the hybrid CNN + Bi-LSTM model:

Step 1: Setting Up the Environment

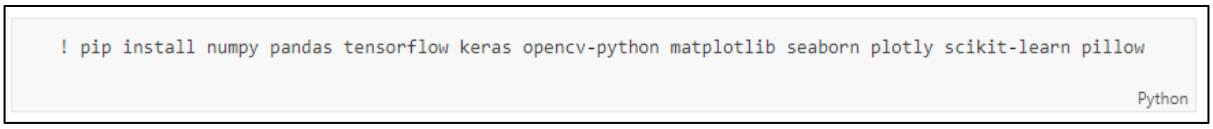
A. Install Anaconda (optional): If you prefer managing environments with Anaconda, install it from [Anaconda's official website] (https://www.anaconda.com/).

B. Create a New Environment:



```
conda create -n brain_tumor_detection python=3.8
conda activate brain_tumor_detection
```

C. Install Required Libraries:



```
! pip install numpy pandas tensorflow keras opencv-python matplotlib seaborn plotly scikit-learn pillow
```

Step 2: Preparing the Dataset

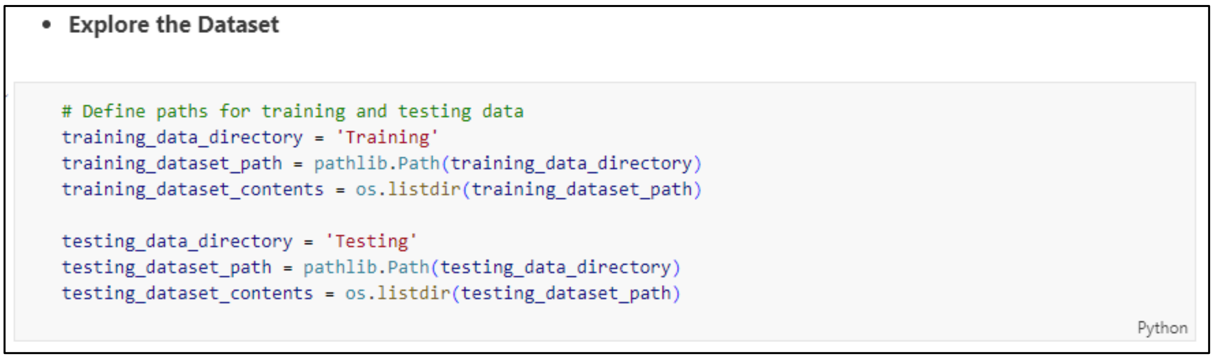
A. Download the dataset from the provided Figshare link.
B. Extract the dataset structure.

Step 3: Running the Code

A. Open Jupyter Notebook or VS Code in the project directory.

B. Run the Code:

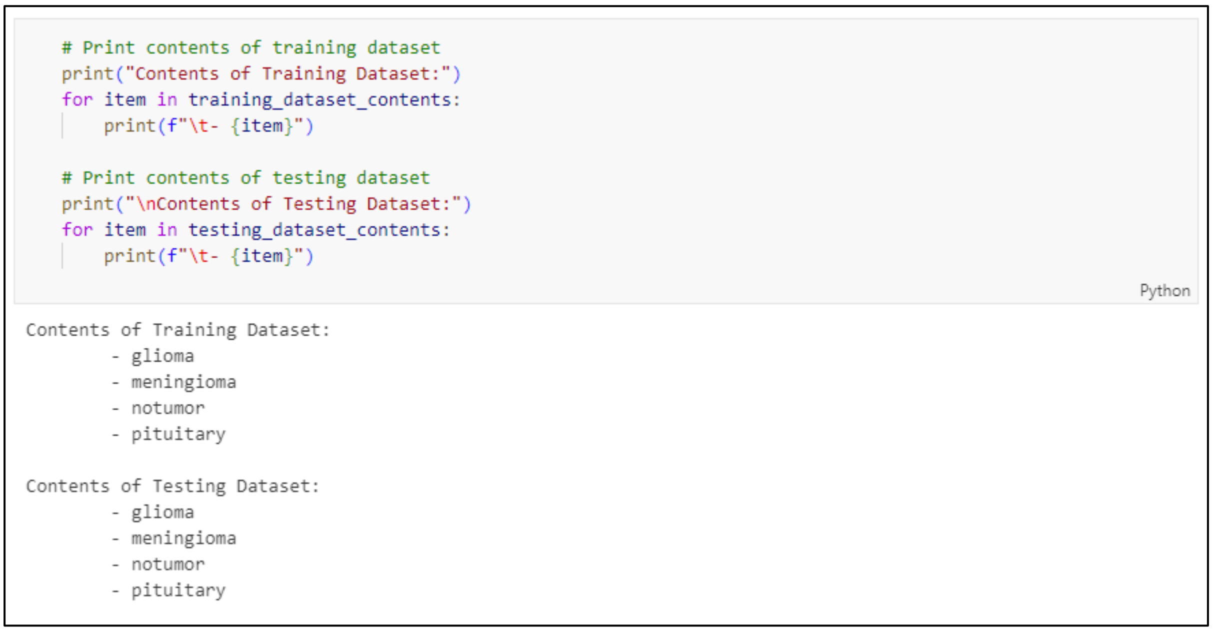
- Start by running the data preparation section to load and preprocess the images.



- **Explore the Dataset**

```python
# Define paths for training and testing data
training_data_directory = 'Training'
training_dataset_path = pathlib.Path(training_data_directory)
training_dataset_contents = os.listdir(training_dataset_path)

testing_data_directory = 'Testing'
testing_dataset_path = pathlib.Path(testing_data_directory)
testing_dataset_contents = os.listdir(testing_dataset_path)
```

Python

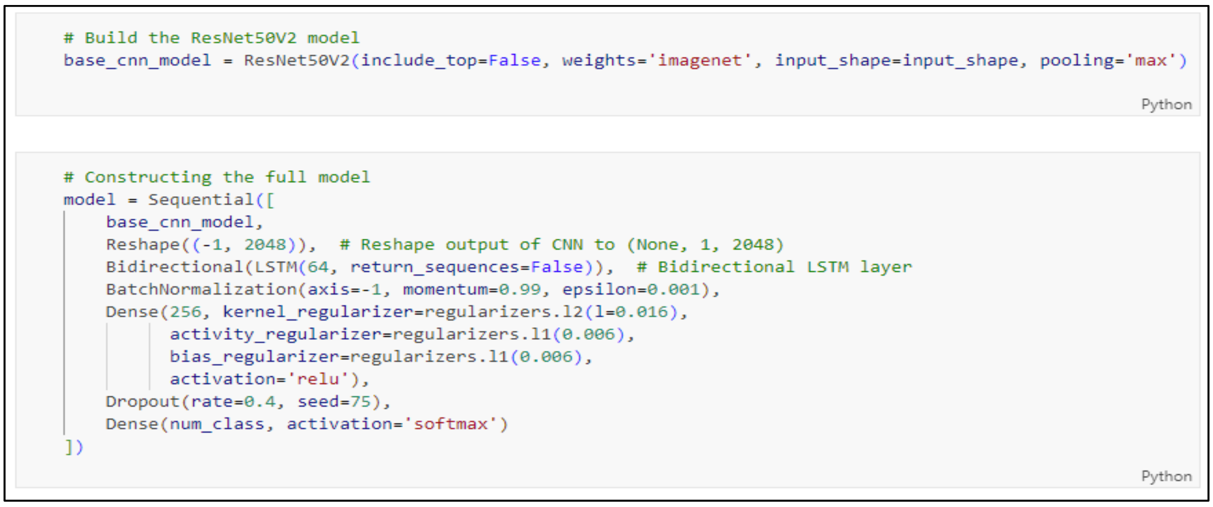


```python
# Print contents of training dataset
print("Contents of Training Dataset:")
for item in training_dataset_contents:
    print(f"\t- {item}")

# Print contents of testing dataset
print("\nContents of Testing Dataset:")
for item in testing_dataset_contents:
    print(f"\t- {item}")
```

Python

```
Contents of Training Dataset:
        - glioma
        - meningioma
        - notumor
        - pituitary

Contents of Testing Dataset:
        - glioma
        - meningioma
        - notumor
        - pituitary
```

- Execute the model-building section to define the ResNet50V2 and Bi-LSTM architecture.



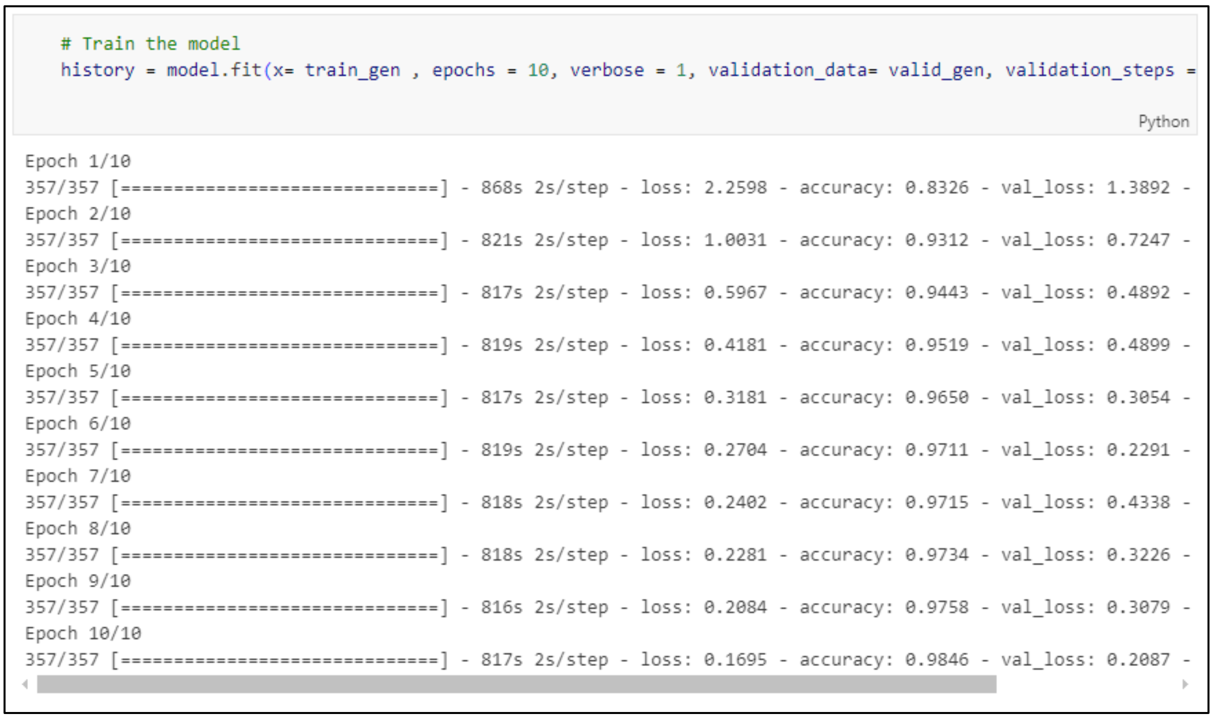
```python
# Build the ResNet50V2 model
base_cnn_model = ResNet50V2(include_top=False, weights='imagenet', input_shape=input_shape, pooling='max')
```

Python

```python
# Constructing the full model
model = Sequential([
    base_cnn_model,
    Reshape((-1, 2048)),  # Reshape output of CNN to (None, 1, 2048)
    Bidirectional(LSTM(64, return_sequences=False)),  # Bidirectional LSTM layer
    BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001),
    Dense(256, kernel_regularizer=regularizers.l2(l=0.016),
          activity_regularizer=regularizers.l1(0.006),
          bias_regularizer=regularizers.l1(0.006),
          activation='relu'),
    Dropout(rate=0.4, seed=75),
    Dense(num_class, activation='softmax')
])
```

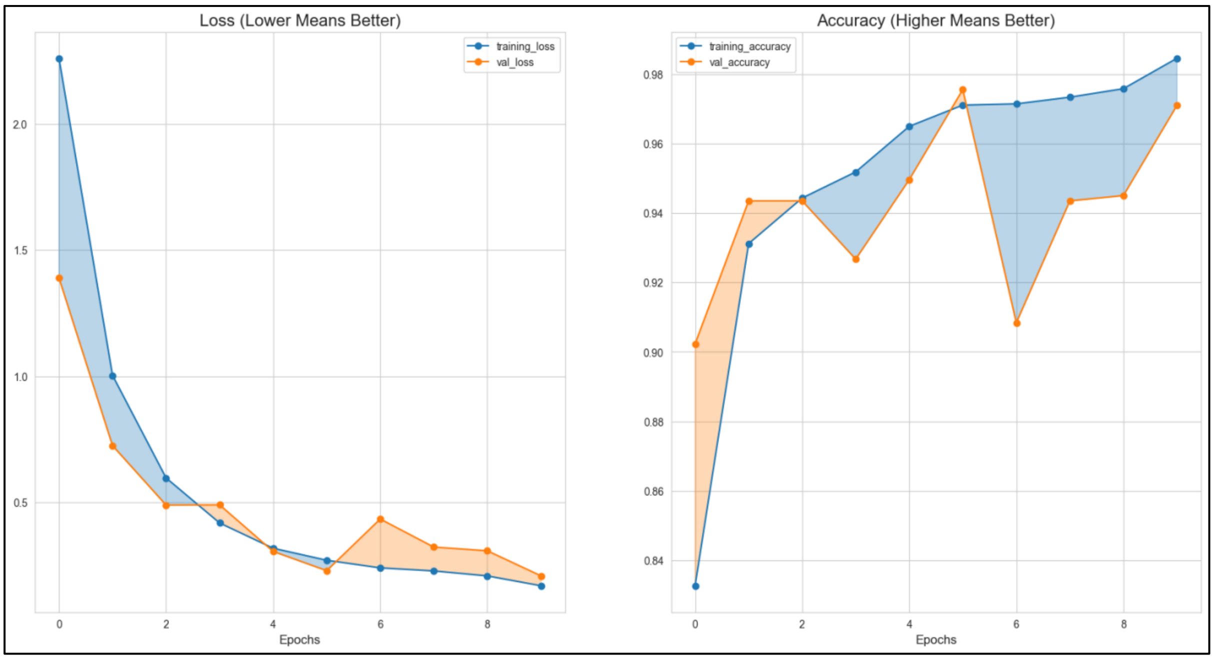
Python

- Train the model by running the training section.

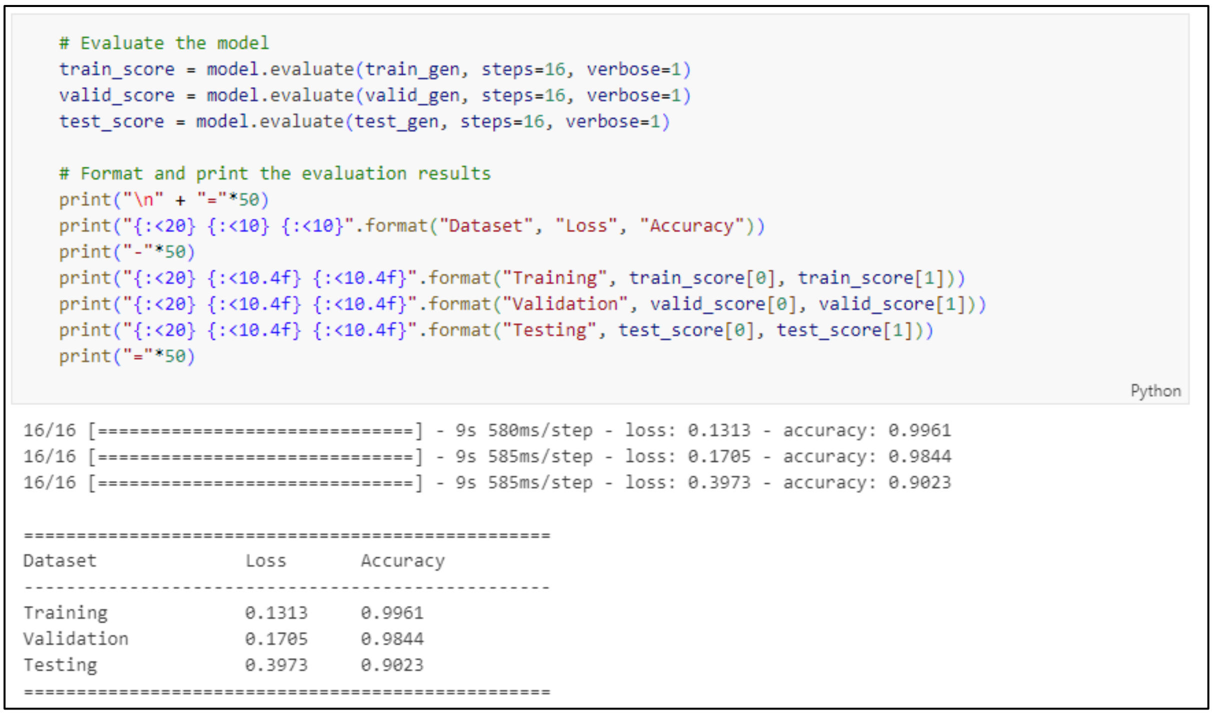


```python
# Train the model
history = model.fit(x= train_gen , epochs = 10, verbose = 1, validation_data= valid_gen, validation_steps =
```

```
Epoch 1/10
357/357 [==============================] - 868s 2s/step - loss: 2.2598 - accuracy: 0.8326 - val_loss: 1.3892 -
Epoch 2/10
357/357 [==============================] - 821s 2s/step - loss: 1.0031 - accuracy: 0.9312 - val_loss: 0.7247 -
Epoch 3/10
357/357 [==============================] - 817s 2s/step - loss: 0.5967 - accuracy: 0.9443 - val_loss: 0.4892 -
Epoch 4/10
357/357 [==============================] - 819s 2s/step - loss: 0.4181 - accuracy: 0.9519 - val_loss: 0.4899 -
Epoch 5/10
357/357 [==============================] - 817s 2s/step - loss: 0.3181 - accuracy: 0.9650 - val_loss: 0.3054 -
Epoch 6/10
357/357 [==============================] - 819s 2s/step - loss: 0.2704 - accuracy: 0.9711 - val_loss: 0.2291 -
Epoch 7/10
357/357 [==============================] - 818s 2s/step - loss: 0.2402 - accuracy: 0.9715 - val_loss: 0.4338 -
Epoch 8/10
357/357 [==============================] - 818s 2s/step - loss: 0.2281 - accuracy: 0.9734 - val_loss: 0.3226 -
Epoch 9/10
357/357 [==============================] - 816s 2s/step - loss: 0.2084 - accuracy: 0.9758 - val_loss: 0.3079 -
Epoch 10/10
357/357 [==============================] - 817s 2s/step - loss: 0.1695 - accuracy: 0.9846 - val_loss: 0.2087 -
```



- Evaluate the model's performance and visualise the results.



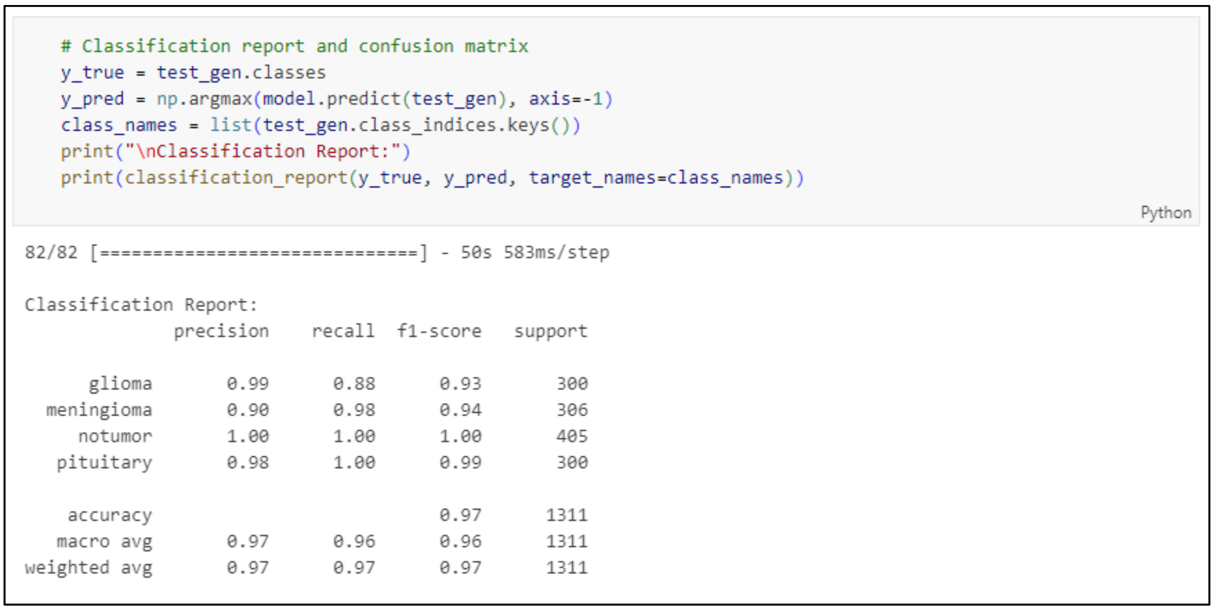
```python
# Evaluate the model
train_score = model.evaluate(train_gen, steps=16, verbose=1)
valid_score = model.evaluate(valid_gen, steps=16, verbose=1)
test_score = model.evaluate(test_gen, steps=16, verbose=1)

# Format and print the evaluation results
print("\n" + "="*50)
print("{:<20} {:<10} {:<10}".format("Dataset", "Loss", "Accuracy"))
print("-"*50)
print("{:<20} {:<10.4f} {:<10.4f}".format("Training", train_score[0], train_score[1]))
print("{:<20} {:<10.4f} {:<10.4f}".format("Validation", valid_score[0], valid_score[1]))
print("{:<20} {:<10.4f} {:<10.4f}".format("Testing", test_score[0], test_score[1]))
print("="*50)
```

Python

```
16/16 [==============================] - 9s 580ms/step - loss: 0.1313 - accuracy: 0.9961
16/16 [==============================] - 9s 585ms/step - loss: 0.1705 - accuracy: 0.9844
16/16 [==============================] - 9s 585ms/step - loss: 0.3973 - accuracy: 0.9023

==================================================
Dataset              Loss       Accuracy
--------------------------------------------------
Training             0.1313     0.9961
Validation           0.1705     0.9844
Testing              0.3973     0.9023
==================================================
```

Step 4: Evaluating the Model

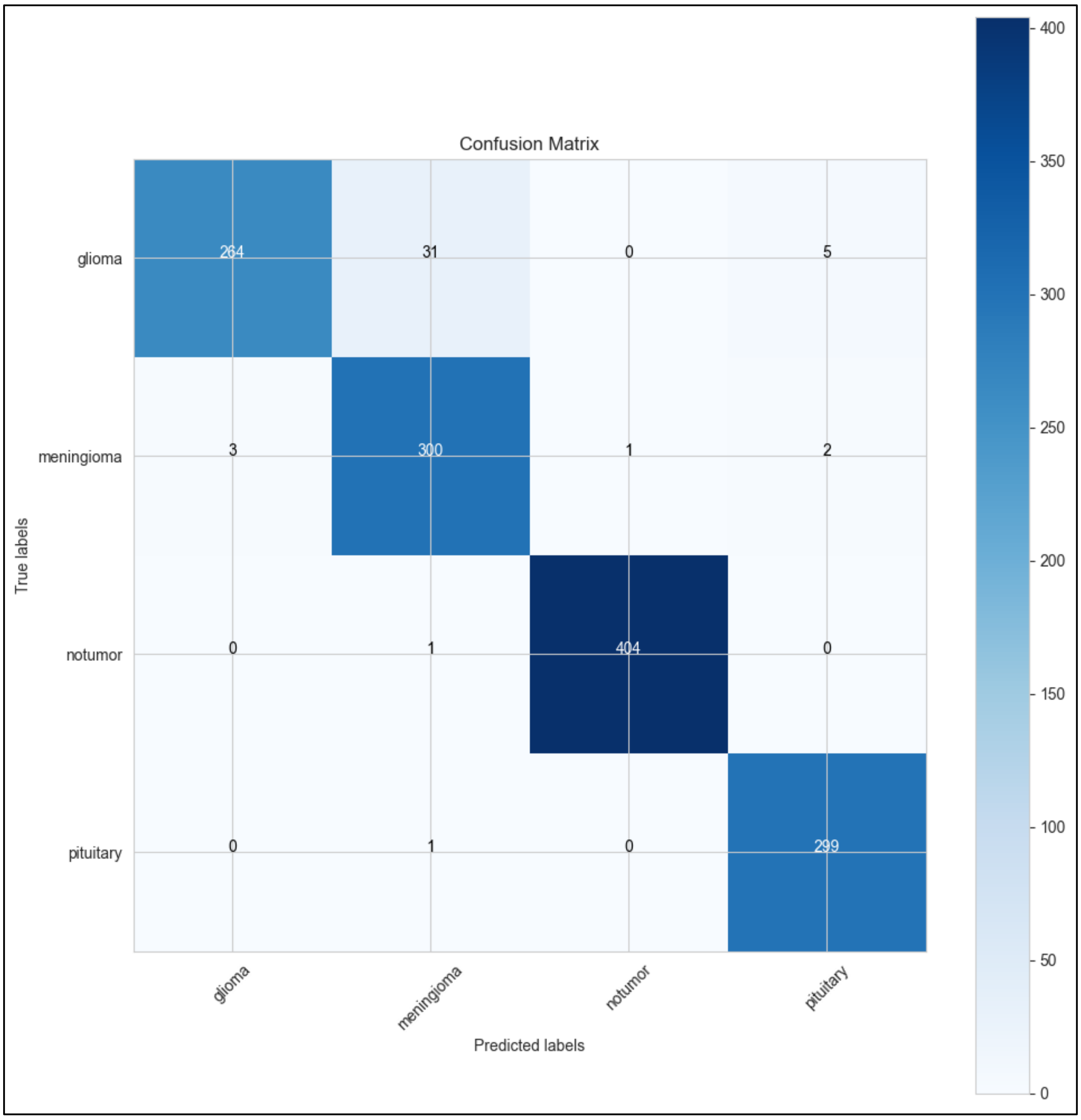
- After training, assess the model's accuracy using the validation and test datasets.
- Generate and review the confusion matrix and classification report for detailed performance metrics.



```python
# Classification report and confusion matrix
y_true = test_gen.classes
y_pred = np.argmax(model.predict(test_gen), axis=-1)
class_names = list(test_gen.class_indices.keys())
print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=class_names))
```

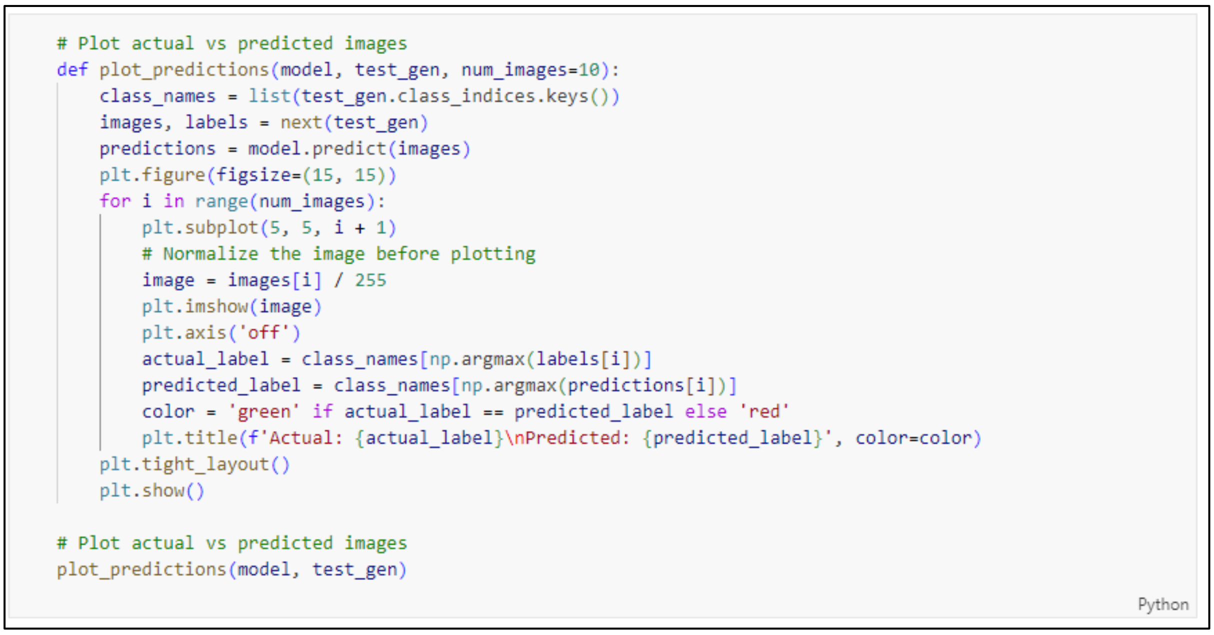
Python

```
82/82 [==============================] - 50s 583ms/step

Classification Report:
              precision    recall  f1-score   support

      glioma       0.99      0.88      0.93       300
  meningioma       0.90      0.98      0.94       306
     notumor       1.00      1.00      1.00       405
   pituitary       0.98      1.00      0.99       300

    accuracy                           0.97      1311
   macro avg       0.97      0.96      0.96      1311
weighted avg       0.97      0.97      0.97      1311
```

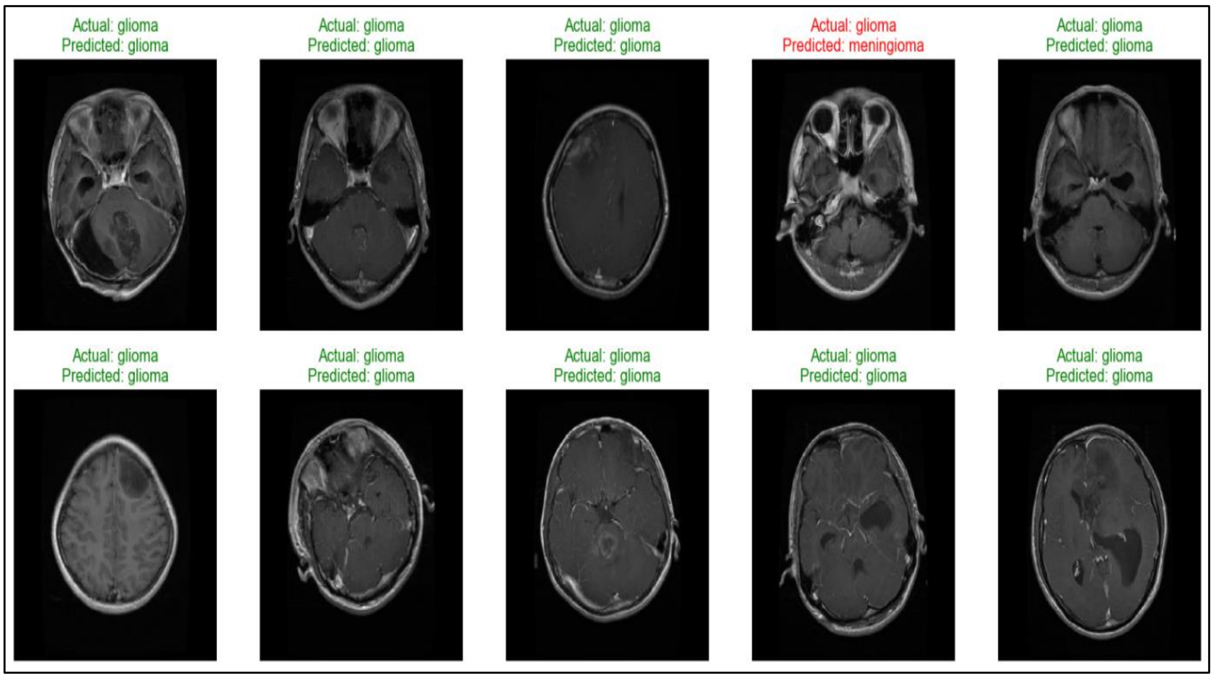


Step 5: Making Predictions

- Use the provided functions to visualise the model's predictions on test images, comparing actual and predicted labels.



```python
# Plot actual vs predicted images
def plot_predictions(model, test_gen, num_images=10):
    class_names = list(test_gen.class_indices.keys())
    images, labels = next(test_gen)
    predictions = model.predict(images)
    plt.figure(figsize=(15, 15))
    for i in range(num_images):
        plt.subplot(5, 5, i + 1)
        # Normalize the image before plotting
        image = images[i] / 255
        plt.imshow(image)
        plt.axis('off')
        actual_label = class_names[np.argmax(labels[i])]
        predicted_label = class_names[np.argmax(predictions[i])]
        color = 'green' if actual_label == predicted_label else 'red'
        plt.title(f'Actual: {actual_label}\nPredicted: {predicted_label}', color=color)
    plt.tight_layout()
    plt.show()

# Plot actual vs predicted images
plot_predictions(model, test_gen)
```



This configuration manual should guide you through the setup and execution of your research project. If any issues arise during the process, ensure that all dependencies are correctly installed, and that the dataset is properly organised.

References

Python: https://www.python.org
Dataset Link: Brain Tumor MRI Dataset