National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Yamuna Sai Penumudi
Student ID: x22174851

School of Computing
National College of Ireland

Supervisor: Abdul Shahid

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Yamuna Sai Penumudi……

Student ID: x22174851……

Programme: Master of Science in Data Analytics …… **Year:** 2023-24……

Module: MSc Research Project……

Lecturer: Abdul Shahid……

Submission Due Date: April 25th 2024……

Project Title: Precision Medicine in Neurology: In-depth Investigation and Revolutionizing Brain Tumor Detection and Treatment

Word Count: ……436…… **Page Count:** ……12……..…

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Yamuna Sai Penumudi

Date: 25-04-2024

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

Yamuna Sai Penumudi
x22174851

1. Introduction

This manual demonstrates all the instructions on setting up and executing the code for the code implementation of In depth Investigation and Revolutionizing Brain Tumor Detection and Treatment. The application is implemented in Python and incorporates advanced neural network method approaches. The following sections guide through the necessary requirements configurations and tools.

2. System Specification

The classification recommendation system has been developed on these following hardware configurations:

- Process: Intel i7 generation
- Operating System: Windows 11 (Home)
- Ram: 16 GB (DDR4)
- Stroage Hard Drive: 512GB (SSD)

3. Softwares Used:

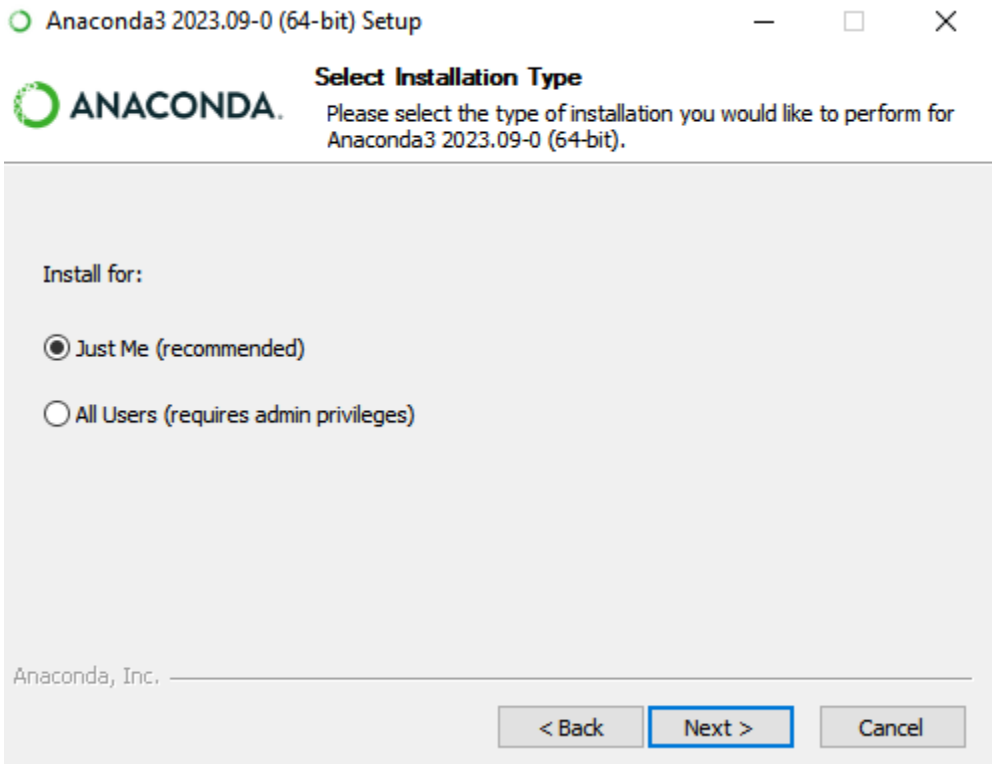
The following tools which are required to use and development for brain tumor detection and classification system:

- Anaconda
- Tensorflow and Keras
- Pandas
- Numpy
- Matplotlib
- Seaborn
- Sklearn
- Jupyter

4. Installation of the Software:

- First download the anaconda from their official website and then start installing to the operation system website: https://www.anaconda.com

- Chosen it for (Just Me) and then clicked on Next until the installation get started.



- Creates the new virtual environment for the purpose of the application (Brain Tumor Detection System)

```
PS D:\new\assignment_left\application> virtualenv brain_tumor_detection
created virtual environment CPython3.11.4.final.0-64 in 4379ms
  creator CPython3Windows(dest=D:\new\assignment_left\application\brain_tumor_detection, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\rohit\AppData\Local\pypa\virtualenv)
    added seed packages: pip==24.0, setuptools==69.1.1, wheel==0.42.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
PS D:\new\assignment_left\application>
```

- Activate the new virtual environment and install the required packages to make the research get done by necessary packages.

5. Source of Dataset

Gather the MRI Scan images dataset which would be suitable for training for deep learning based neural networks model. Datasets contains the various types of brain tumor mri scan images on platforms where but I used the Kaggle to choose the dataset.

6. Code Execution

Open the jupyter notebook to start developing or modifying the. ipynb (Integrated Python Notebook) for the task from the beginning loading the dataset to evaluating the models.

This notebook is focused on precision medicine in neurology, specifically investigating and revolutionizing brain tumor detection and treatment using deep learning techniques. The goal is to build a model that can accurately classify brain tumor MRI images into different categories, enabling early and accurate diagnosis of brain tumors.

The notebook will cover the following key steps:

1. **Data Preparation:**
   - Load and preprocess the brain tumor dataset.
   - Perform exploratory data analysis (EDA) to understand the dataset.
2. **Model Building:**
   - Build a deep learning model using the EfficientNetB3 architecture for image classification.
   - Compile the model with appropriate loss and optimizer.
3. **Model Training and Evaluation:**
   - Train the model on the training dataset.
   - Evaluate the model on the validation and test datasets.
   - Plot training curves to visualize the training and validation performance.
4. **Model Testing:**
   - Test the trained model on sample images from the test dataset.
   - Save the trained model for future use.
5. **Prediction:**
   - Use the trained model to predict the class of custom brain tumor MRI images.

By the end of this notebook, we aim to develop a robust deep learning model for brain tumor detection, which can potentially contribute to advancing precision medicine in neurology.

Execution to Run the File:

- Import the required libraries.

1. Imports the Required Libraries

```
# For Data Processing
import numpy as np
import pandas as pd
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from PIL import Image, ImageEnhance

# For ML Models
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D , MaxPooling2D , Flatten , Activation , Dense , Dropout , BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam , Adamax
from tensorflow.keras import regularizers

# For Data Visualization
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
import plotly.express as px

# Miscellaneous
import os
import pathlib
import itertools
```

- Load the dataset.

```
# Function to create dataframe from data_path
def create_dataframe(data_path):
    # Initialize empty lists for file paths and labels
    filepaths = []
    labels = []

    # Get list of subdirectories (folds) in data_path
    folds = os.listdir(data_path)

    # Iterate over each subdirectory (fold)
    for fold in folds:
        # Construct full path to the current fold
        fold_path = os.path.join(data_path, fold)
        # Get list of files in the current fold
        filelists = os.listdir(fold_path)

        # Iterate over each file in the fold
        for file in filelists:
            # Construct full path to the file
            filepaths.append(os.path.join(fold_path, file))
            # Assign label based on fold name
            labels.append(fold)

    # Create pandas Series for filepaths and labels
    filepaths_series = pd.Series(filepaths, name='filepaths')
    labels_series = pd.Series(labels, name='label')

    # Concatenate filepaths and labels Series into a DataFrame
    dataframe = pd.concat([filepaths_series, labels_series], axis=1)
    return dataframe
```

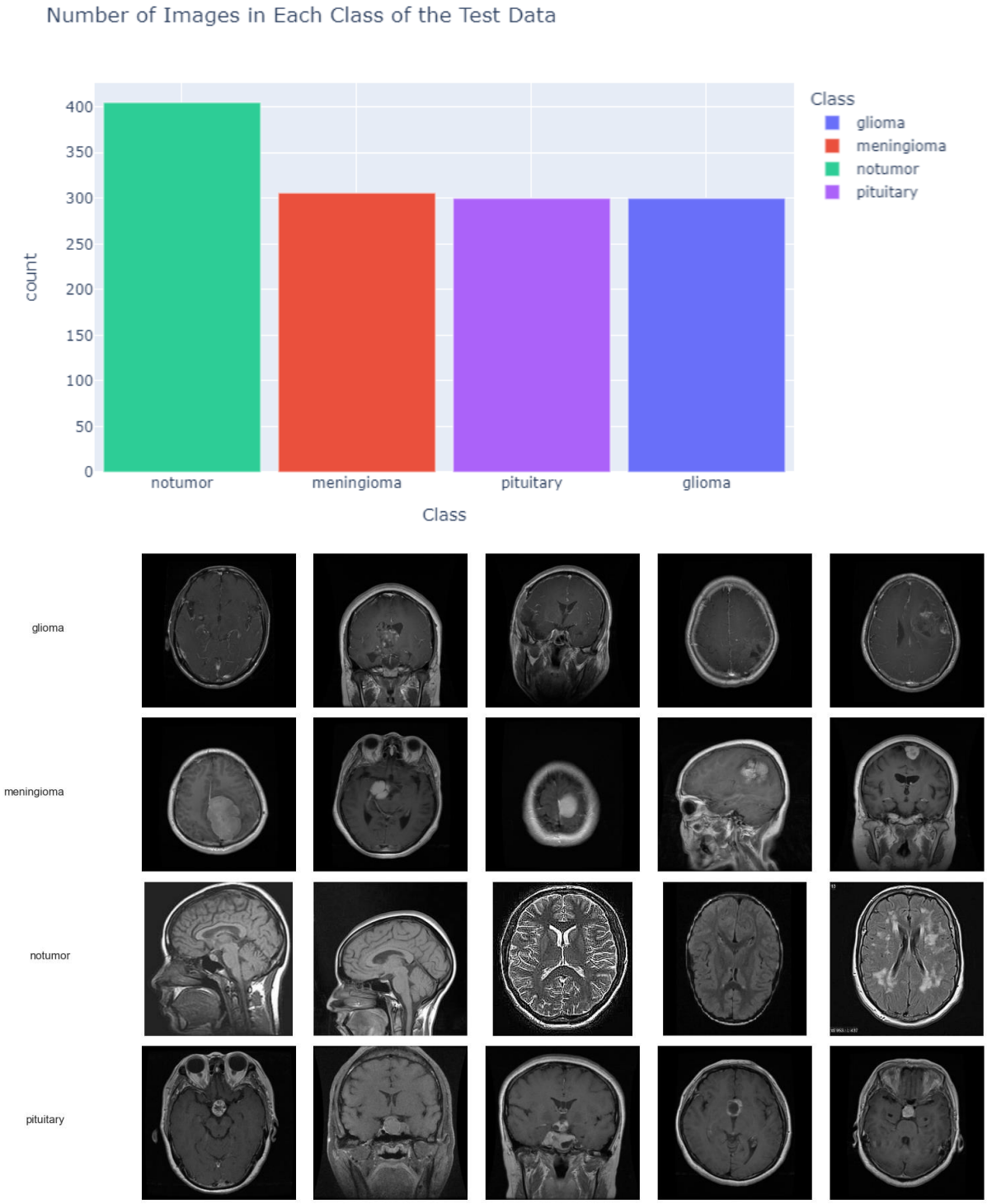
```
# Define paths for training and testing data
train_data_path = 'Training'
train_dataset_path = pathlib.Path(train_data_path)
train_dataset_contains = os.listdir(train_dataset_path)
print("Training Dataset Contains: ",*train_dataset_contains,sep='\n\t\t\t')

test_data_path = 'Testing'
test_dataset_path = pathlib.Path(test_data_path)
test_dataset_contains = os.listdir(test_dataset_path)
print("\nTesting Dataset Contains: ",*test_dataset_contains,sep='\n\t\t\t')
```

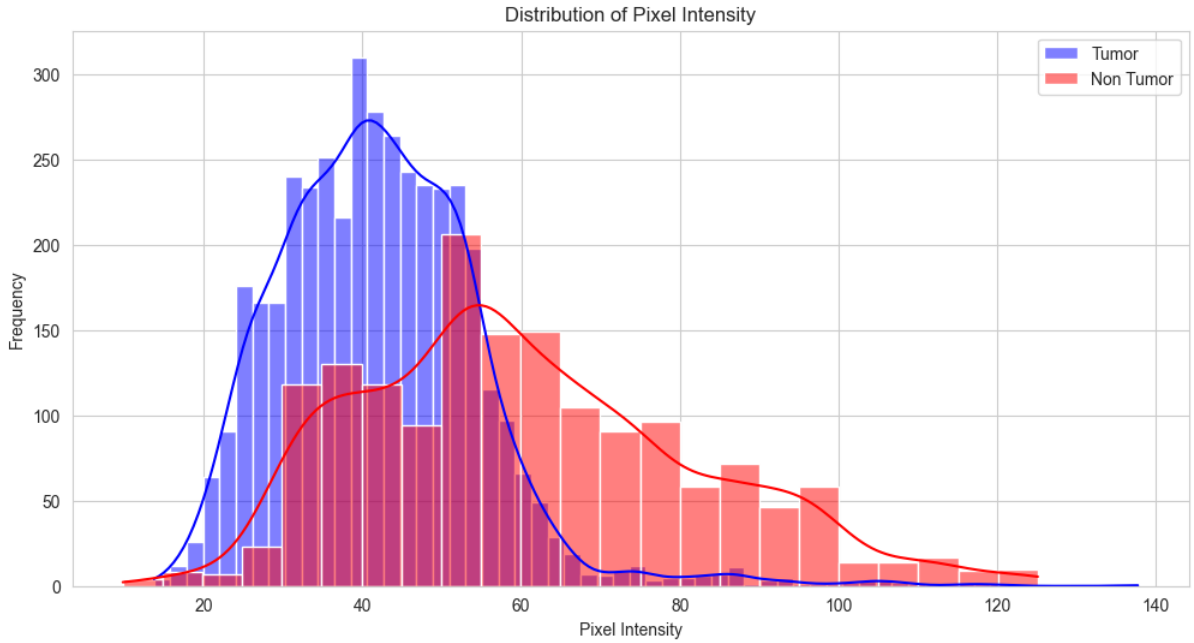
```
Training Dataset Contains:
                        glioma
                        meningioma
                        notumor
                        pituitary

Testing Dataset Contains:
                        glioma
                        meningioma
                        notumor
                        pituitary
```

```
# Create DataFrame for training data
train_df = create_dataframe(train_data_path)
```

```
# Create DataFrame for testing data
test_df = create_dataframe(test_data_path)
```

- Exploratory Data Analysis (EDA)


Number of Images in Each Class of the Test Data
count
400
350
300
250
200
150
100
50
0
notumor
meningioma
pituitary
glioma
Class
Class
glioma
meningioma
notumor
pituitary
glioma
meningioma
notumor
pituitary



- Model Selection which contains the feature selection, splitting of dataset and model initialization and model training.

4. Splitting and Preprocessing of the Brain Tumor Dataset (Preprocess the MRI images (resizing, normalization, etc.).)

```
# Splitting the data into validation and test sets
valid, test = train_test_split(test_df, train_size=0.5, shuffle=True, random_state=42)
```

```
# Image Data Generator setup
img_size = (224, 224)
batch_size = 16
```

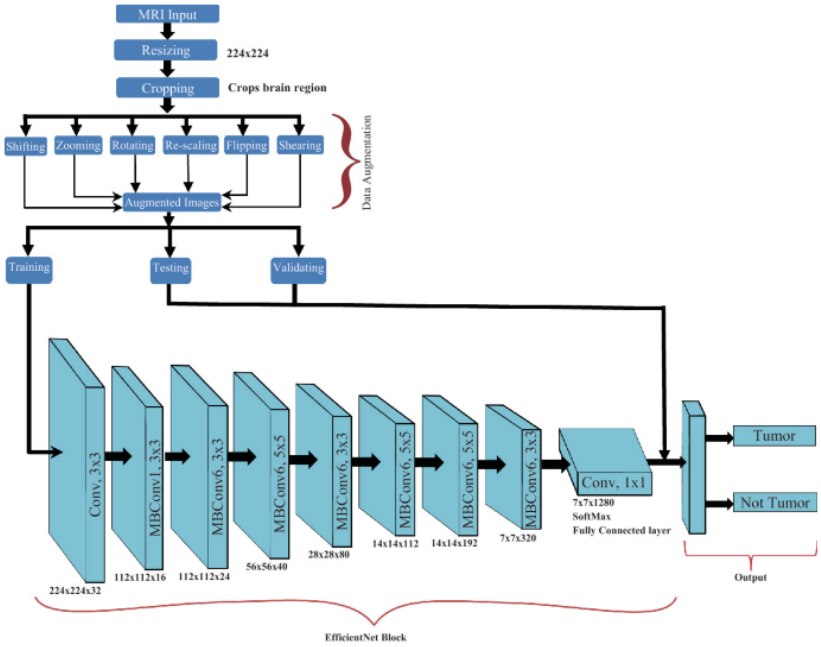
```
# Train Data Generator
tr_gen = ImageDataGenerator()
ts_gen = ImageDataGenerator()
```

```
# Train Data Generator setup
train_gen = tr_gen.flow_from_dataframe(train_df, x_col='filepaths', y_col='label', target_size=img_size,
                                       class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=batch_size)

# Validation Data Generator setup
valid_gen = ts_gen.flow_from_dataframe(valid, x_col='filepaths', y_col='label', target_size=img_size,
                                       class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=batch_size)

# Test Data Generator setup
test_gen = ts_gen.flow_from_dataframe(test, x_col='filepaths', y_col='label', target_size=img_size,
                                      class_mode='categorical', color_mode='rgb', shuffle=False, batch_size=batch_size)
```

```
Found 5712 validated image filenames belonging to 4 classes.
Found 655 validated image filenames belonging to 4 classes.
Found 656 validated image filenames belonging to 4 classes.
```

5. Model Building (EfficientNetB3 model)

EfficientNetB3 is a type of deep learning model that is designed to be very effective at understanding and recognizing images. It's like a super-smart system that learns from lots of examples to recognize things in pictures. The "B3" part refers to the specific size or complexity of the model, with larger numbers indicating more complex models that can potentially understand more detailed features in images. Overall, EfficientNetB3 is known for being efficient in terms of its size and computational requirements, while still being very good at tasks like image classification.




```
# Build the EfficientNetB3 model
base_model = tf.keras.applications.efficientnet.EfficientNetB3(include_top=False, weights='imagenet', input_shape=input_shape, pc

# Constructing the full model
model = Sequential([
    base_model,
    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')
])

# Compile the model
model.compile(Adamax(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

# Display the model summary
print("Model Summary:")
model.summary()
```

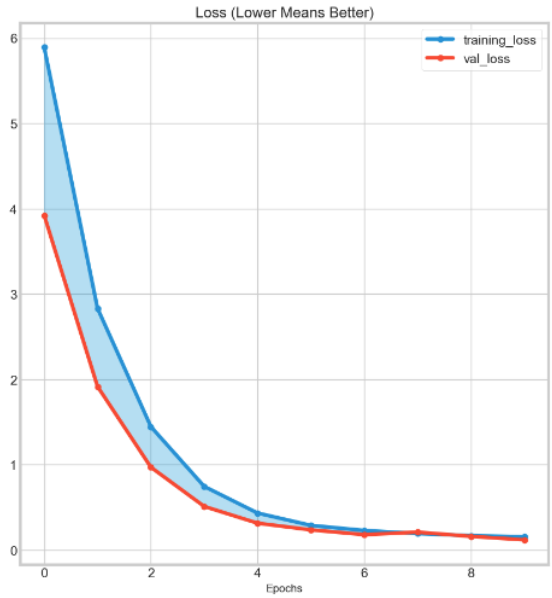
```
Model Summary:
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb3 (Functional  (None, 1536)              10783535
 )

 batch_normalization_2 (Bat  (None, 1536)              6144
 chNormalization)

 dense_4 (Dense)             (None, 256)               393472

 dropout_2 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 4)                 1028

=================================================================
Total params: 11184179 (42.66 MB)
Trainable params: 11093804 (42.32 MB)
Non-trainable params: 90375 (353.03 KB)
_________________________________________________________________
```

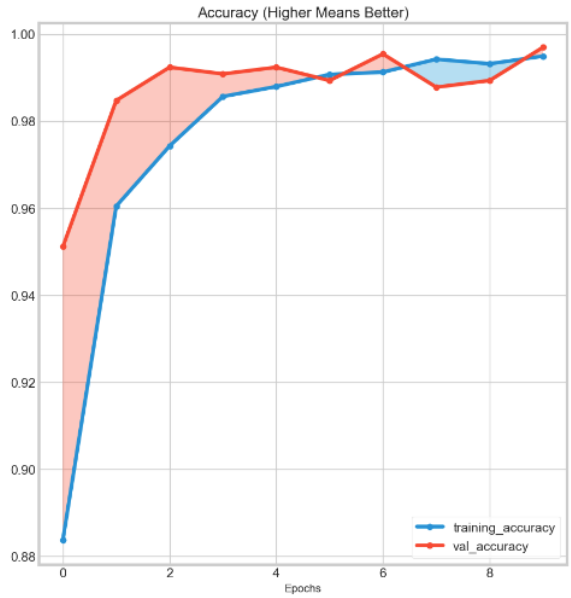
6. Model Training and Evaluation

```
# Train the model with transformers
history = model.fit(x= train_gen , epochs = 10, verbose = 1, validation_data= valid_gen,validation_steps = None , shuffle = False
```

```
Epoch 1/10
357/357 [==============================] - 1108s 3s/step - loss: 5.8919 - accuracy: 0.8838 - val_loss: 3.9216 - val_accuracy:
0.9511
Epoch 2/10
357/357 [==============================] - 1047s 3s/step - loss: 2.8326 - accuracy: 0.9604 - val_loss: 1.9157 - val_accuracy:
0.9847
Epoch 3/10
357/357 [==============================] - 1041s 3s/step - loss: 1.4454 - accuracy: 0.9743 - val_loss: 0.9698 - val_accuracy:
0.9924
Epoch 4/10
357/357 [==============================] - 1056s 3s/step - loss: 0.7449 - accuracy: 0.9856 - val_loss: 0.5104 - val_accuracy:
0.9908
Epoch 5/10
357/357 [==============================] - 1130s 3s/step - loss: 0.4327 - accuracy: 0.9879 - val_loss: 0.3140 - val_accuracy:
0.9924
Epoch 6/10
357/357 [==============================] - 974s 3s/step - loss: 0.2882 - accuracy: 0.9907 - val_loss: 0.2355 - val_accuracy: 0.
9893
Epoch 7/10
357/357 [==============================] - 991s 3s/step - loss: 0.2284 - accuracy: 0.9912 - val_loss: 0.1796 - val_accuracy: 0.
9954
Epoch 8/10
357/357 [==============================] - 1054s 3s/step - loss: 0.1924 - accuracy: 0.9942 - val_loss: 0.2091 - val_accuracy:
0.9878
Epoch 9/10
357/357 [==============================] - 1047s 3s/step - loss: 0.1677 - accuracy: 0.9932 - val_loss: 0.1594 - val_accuracy:
0.9893
Epoch 10/10
357/357 [==============================] - 1092s 3s/step - loss: 0.1546 - accuracy: 0.9949 - val_loss: 0.1190 - val_accuracy:
0.9969
```

- Model Evaluation





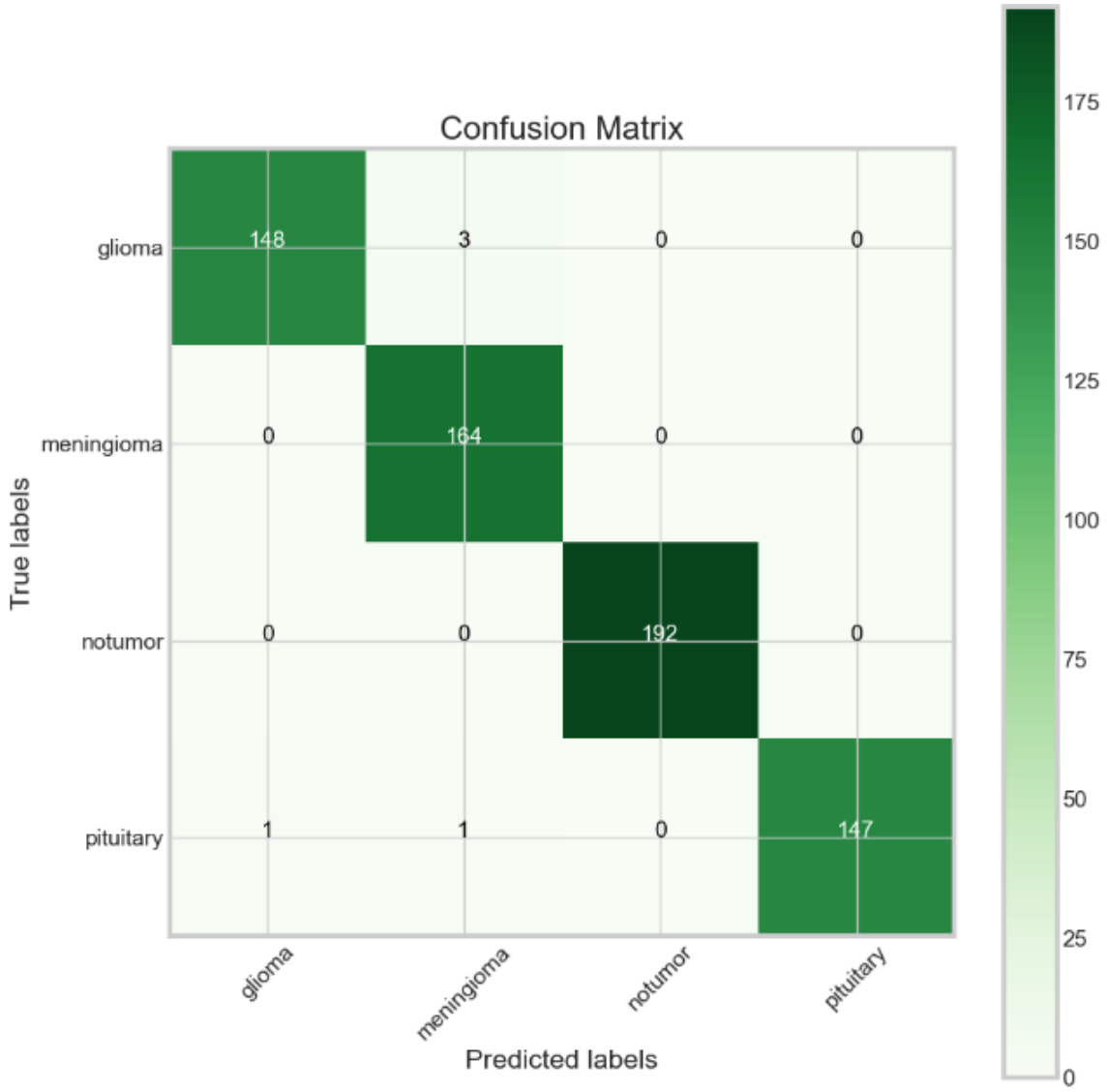
Dataset	Loss	Accuracy
Training	0.1143	1.0000
Validation	0.1264	0.9922
Testing	0.1347	0.9922

This below illustration contains the classification report for EfficientNetB3 mode:

Classification Report:

	precision	recall	f1-score	support
glioma	0.99	0.98	0.99	151
meningioma	0.98	1.00	0.99	164
notumor	1.00	1.00	1.00	192
pituitary	1.00	0.99	0.99	149
accuracy			0.99	656
macro avg	0.99	0.99	0.99	656
weighted avg	0.99	0.99	0.99	656

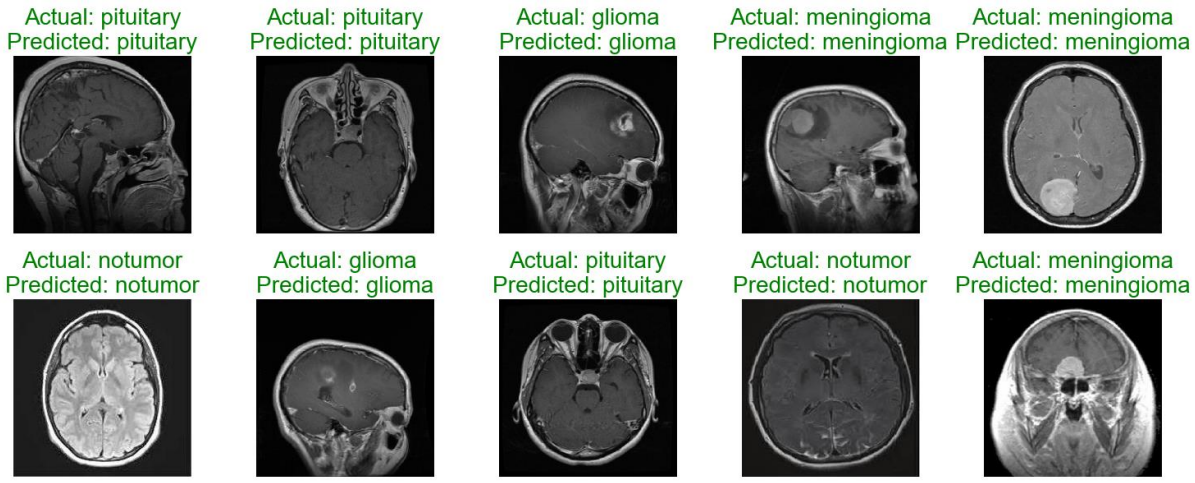
Confusion matrix after evaluation the model prediction



- Model Testing: After the model training and evaluation, there the step comes to test the model through the random detection of brain tumor through the MRI scans.

7. Model Testing

```
# 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)
```



- Saving the Trained Model to use this for the further deployment or for customization brain tumor detection.

```
# Save the trained model
model.save("brain_tumor_model.h5")
```

Python

```
# Load the saved model
loaded_model = tf.keras.models.load_model("brain_tumor_model.h5")
```

Python

This configuration manual provides as a comprehensive exploration for configuring the installation the required softwares or tools to implementation of the code for understanding the brain tumor detection system using deep learning based CNN model of EfficientNetB3 on the Brain Tumors MRI scan images dataset.

References

Anaconda: https://docs.anaconda.com/free/anaconda/install/windows/
Kaggle Dataset Source: https://www.kaggle.com/