National College of Ireland

Configuration Manual

MSc Research Project
Msc. in Data Analytics

Niladri Sekhar Bandyopadhyay
Student ID: X22218289

School of Computing
National College of Ireland

Supervisor: Qurrat Ul Ain

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Niladri Sekhar Bandyopadhyay
Student ID:	X22218289
Programme:	Msc. in Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Qurrat Ul Ain
Submission Due Date:	08/08/2024
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	6

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:	
Date:	15th September 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

Niladri Sekhar Bandyopadhyay
X22218289

1 Computer Configuration

This Current research has been done on a laptop with a configuration with

Processor - Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz
Ram - 16.0 GB (15.9 GB usable)
System Type - 64-bit operating system, X64-based processor
Operating System - Windows 11 Home.

2 Tool Configuration

This Current research on Brain tumor detection has been implemented using tools like Jupyter Notebook.The following figure Figure 1 is the Jupyter and python version form the terminal

```
(base) C:\Users\NILADRI>python --version
Python 3.9.13

(base) C:\Users\NILADRI>jupyter --version
Selected Jupyter core packages...
IPython          : 7.31.1
ipykernel        : 6.15.2
ipywidgets       : 7.6.5
jupyter_client   : 7.3.4
jupyter_core     : 4.11.1
jupyter_server   : 1.18.1
jupyterlab       : 3.4.4
nbclient         : 0.5.13
nbconvert        : 6.4.4
nbformat         : 5.5.0
notebook         : 6.4.12
qtconsole        : 5.2.2
traitlets        : 5.1.1

(base) C:\Users\NILADRI>
```

Figure 1: Tools Version

3 Libraries to install

This Third Section is about the libraries to install means which libraries are the pre-requisite to install before running the main brain tumor Jupyter Notebook file, and those are the following:

Numpy, Pandas, MatplotLib, seaborn, OpenCV, Pillow, SciPy, Scikit-learn, TensorFlow, Keras, Segmentation Models, SimpleITK

First, check whether these libraries are installed on that particular system or not where the brain tumor detection Jupyter Notebook will be running if not installed then by running the command in the terminal one can install those files in the system. Execute the below command in the terminal in case any of those libraries are not installed

pip install Numpy pandas matplotlib seaborn OpenCV-python Pillow SciPy Scikit-learn TensorFlow keras segmentation-models SimpleITK

in case some version issue comes into the picture then first use the command **conda install conda=version** and then again install all the libraries. After then if anything comes into the picture then create a virtual new environment by putting the command **conda create –name myenv python=3.9** Here in this command replace myenv name with the desired environment name

4 Folder Information

Before executing the main Jupyter Notebook file make sure that all the libraries are installed successfully in the system and system configuration is maintained properly.

Now, this project needs a folder called Research Project and in that particular folder all the dataset as well as the Python notebook have to be stored clearly seen in Figure 2.

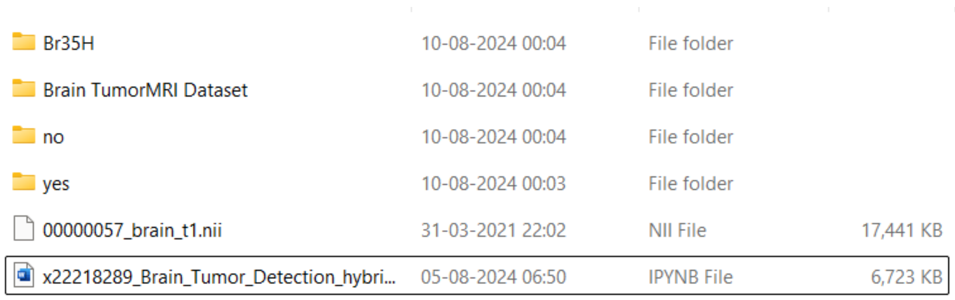


Figure 2: Folder image with dataset

From Figure 2 one can see that Brain tumor Internal data as well as external data is present in this particular folder. Now before running the file path generalization should be done which, has been done in this research by os.chdir('C:/Users/NILADRI/Desktop/Important Folders/Research Project') instead of this one can include their path. Figure 3 is about how to reset the path

```
print(os.getcwd())  # Print current working directory
os.chdir('C:/Users/NILADRI/Desktop/Important Folders/Research Project')
predict_image('./Brain TumorMRI Dataset/Testing/meningioma/Te-me_0013.jpg')
predict_image('./Brain TumorMRI Dataset/Testing/notumor/Te-no_0010.jpg')
```

Figure 3: Reset Path for predicting Image

5 How to Execute the main file

However, if one goes to the kernel in Jupyter Notebook and clicks on Restart, and runs all the code at that time, all the code will be executed, these below are the important steps to execute to predict brain tumor.

5.1 Unzip the BR35H folder

In Figure 4 Unzip the BR35H folder

```
import zipfile
import os

def extract_zip_in_same_location(zip_path):
    # Check if the provided path is a valid zip file
    if not zipfile.is_zipfile(zip_path):
        print(f"{zip_path} is not a valid zip file.")
        return

    # Get the directory where the zip file is located
    zip_dir = os.path.dirname(zip_path)

    # Extract the zip file
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(zip_dir)

    print(f"Extracted {zip_path} to {zip_dir}")

# Example usage
zip_file_path = 'Br35H.zip'
extract_zip_in_same_location(zip_file_path)
```

```
Extracted Br35H.zip to
```

Figure 4: Unzip the BR35H folder

5.2 Data Collection

In Figure 5 Data has been Collected

5.3 Data Augmentation

In Figure 6 Data Augmentation has been implemented

5.4 Split into Train and Test

In Figure 7 Splitting data into Train and Test

5.5 Model Training

In Figure 8 Model Training has been developed

```python
import os

# Define the base directory for the images
base_image_dir = ''

# Create paths for each category of images
healthy_image_path = os.path.join(base_image_dir, 'no/')
tumor_image_path = os.path.join(base_image_dir, 'yes/')

# List images in each category
healthy_image_list = os.listdir(healthy_image_path)
tumor_image_list = os.listdir(tumor_image_path)

# Display the count of images in each category
print(f'Number of images with no tumor: {len(healthy_image_list)}')
print(f'Number of images with tumor: {len(tumor_image_list)}')
```

```
Number of images with no tumor: 1500
Number of images with tumor: 1500
```

Figure 5: Data Collection

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from scipy.ndimage import gaussian_filter, sobel
import numpy as np

# Convert dataset and labels to numpy arrays
image_data_array = np.array(image_data)
image_labels_array = np.array(image_labels)

# Data augmentation and normalization for training
train_data_augmentor  = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=15,  # Rotation within ±15 degrees
    width_shift_range=0.2,  # Translation along the x-axis
    height_shift_range=0.2,  # Translation along the y-axis
    shear_range=0.2,  # Shear transformation
    zoom_range=0.2,  # Scaling
    horizontal_flip=True,  # Horizontal flipping
    vertical_flip=True,  # Vertical flipping
    brightness_range=[0.8, 1.2],  # Brightness adjustment
    preprocessing_function=add_gaussian_noise,
    fill_mode='nearest',  # Fill mode for transformed images
    validation_split=0.2  # Split for validation
)
```

Figure 6: Data Augmentation

```python
import numpy as np
from sklearn.model_selection import train_test_split

# Transform image data into a NumPy array
image_data_np = np.asarray(image_data)

# Split the dataset into training and testing sets
train_images, test_images, train_labels, test_labels = train_test_split(image_data_np, image_labels_array, test_size=0.2, random_

# Normalize the image data to the range [0, 1]
train_images = train_images / 255.0
test_images = test_images / 255.0

# Display the shapes of the datasets
print('Training data shape: ', train_images.shape)
print('Testing data shape: ', test_images.shape)
print('Training labels shape: ', train_labels.shape)
print('Testing labels shape: ', test_labels.shape)
```

```
Training data shape:  (2400, 224, 224, 3)
Testing data shape:  (600, 224, 224, 3)
Training labels shape:  (2400,)
Testing labels shape:  (600,)
```

Figure 7: Split into Train and Test

5.6 Model Prediction

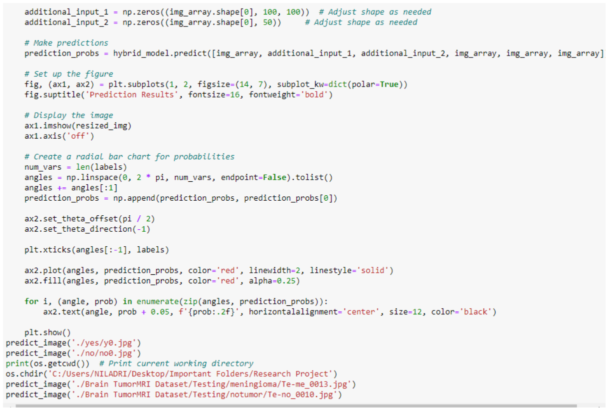
In Figure 9 Developed Model has been Predicted

```python
inception_output = Dense(32, activation='relu', kernel_regularizer=l2(0.01))(inception_output)  # Reduced units

# Concatenate outputs from all models
concatenated = concatenate([model1_output, model2_output, model3_output, unet_output, resnet_output, inception_output])
concatenated = Dropout(0.7)(concatenated)  # Increased dropout rate

# Final Dense Layer for classification
output = Dense(2, activation='softmax')(concatenated)

# Create the hybrid model
hybrid_model = Model(inputs=[input_cnn, input_rnn, input_dense, input_unet, input_resnet, input_inception], outputs=output)

# Compile the model
hybrid_model.compile(optimizer=Adam(learning_rate=1e-5), loss='categorical_crossentropy', metrics=['accuracy'])

# Print model summary
hybrid_model.summary()

# Callbacks
early_stop = EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=6, verbose=1)
checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True, verbose=1)
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard = TensorBoard(log_dir=log_dir, histogram_freq=1)

def scheduler(epoch, lr):
    if epoch < 5:
        return lr
    else:
        return lr * 0.95  # Reduced decay rate

lr_scheduler = LearningRateScheduler(scheduler)

# Train the model with the specified callbacks
training_history = hybrid_model.fit(
    [train_images, np.zeros((train_images.shape[0], 100, 100)), np.zeros((train_images.shape[0], 50)), train_images, train_image
    encoded_train_labels,
    batch_size=16,
    epochs=5,
    validation_data=([test_images, np.zeros((test_images.shape[0], 100, 100)), np.zeros((test_images.shape[0], 50)), test_images
    shuffle=True,
    verbose=1,
    callbacks=[early_stop, reduce_lr, checkpoint, tensorboard, lr_scheduler]
)
```

```
Segmentation Models: using `tf.keras` framework.
Model: "model_2"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	[]
conv2d (Conv2D)	(None, 222, 222, 16)	448	['input_1[0][0]']
batch_normalization (Batch Normalization)	(None, 222, 222, 16)	64	['conv2d[0][0]']
input_4 (InputLayer)	[(None, 224, 224, 3)]	0	[]
input_6 (InputLayer)	[(None, 224, 224, 3)]	0	[]
input_8 (InputLayer)	[(None, 224, 224, 3)]	0	[]
max_pooling2d (MaxPooling2 D)	(None, 111, 111, 16)	0	['batch_normalization[0][0]']
input_2 (InputLayer)	[(None, 100, 100)]	0	[]
input_3 (InputLayer)	[(None, 50)]	0	[]
model_1 (Functional)	(None, 224, 224, 1)	2445615 4	['input_4[0][0]']

Figure 8: Model Training



```python
    additional_input_1 = np.zeros((img_array.shape[0], 100, 100))  # Adjust shape as needed
    additional_input_2 = np.zeros((img_array.shape[0], 50))     # Adjust shape as needed

    # Make predictions
    prediction_probs = hybrid_model.predict([img_array, additional_input_1, additional_input_2, img_array, img_array, img_array]

    # Set up the figure
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7), subplot_kw=dict(polar=True))
    fig.suptitle('Prediction Results', fontsize=16, fontweight='bold')

    # Display the image
    ax1.imshow(resized_img)
    ax1.axis('off')

    # Create a radial bar chart for probabilities
    num_vars = len(labels)
    angles = np.linspace(0, 2 * pi, num_vars, endpoint=False).tolist()
    angles += angles[:1]
    prediction_probs = np.append(prediction_probs, prediction_probs[0])

    ax2.set_theta_offset(pi / 2)
    ax2.set_theta_direction(-1)

    plt.xticks(angles[:-1], labels)

    ax2.plot(angles, prediction_probs, color='red', linewidth=2, linestyle='solid')
    ax2.fill(angles, prediction_probs, color='red', alpha=0.25)

    for i, (angle, prob) in enumerate(zip(angles, prediction_probs)):
        ax2.text(angle, prob + 0.05, f'{prob:.2f}', horizontalalignment='center', size=12, color='black')

    plt.show()
predict_image('./yes/y0.jpg')
predict_image('./no/no0.jpg')
print(os.getcwd())  # Print current working directory
os.chdir('C:/Users/NILADRI/Desktop/Important Folders/Research Project')
predict_image('./Brain TumorMRI Dataset/Testing/meningioma/Te-me_0013.jpg')
predict_image('./Brain TumorMRI Dataset/Testing/notumor/Te-no_0010.jpg')
```

Figure 9: Model Prediction Snippet

In Figure 10 Showing its a brain tumor or not

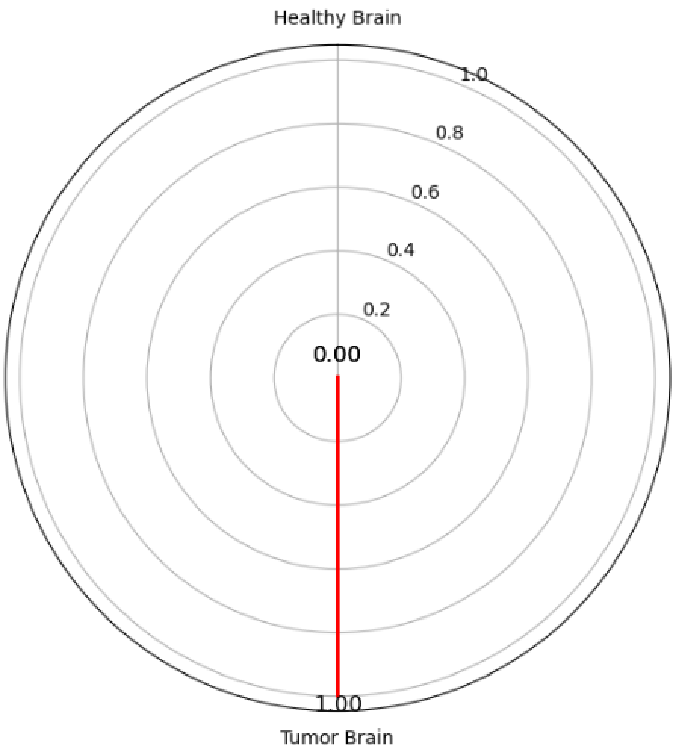


Figure 10: Model Prediction