National College of Ireland

Configuration Manual

MSc Research Project
Artificial Intelligence

Rajratan Gajbhiye
Student ID: x22179038

School of Computing
National College of Ireland

Supervisor: Prof. Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Rajratan Gajbhiye
Student ID:	x22179038
Programme:	Artificial Intelligence
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Prof. Rejwanul Haque
Submission Due Date:	31/01/2024
Project Title:	Configuration Manual
Word Count:	319
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:	Rajratan Laxminarayan Gajbhiye
Date:	30th January 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

Rajratan Gajbhiye
x22179038

1 Introduction

This document provides all the information about the necessary hardware and software requirements for the smooth operation of an advanced deep-learning system designed to recognize handwritten Devanagari characters.

2 Hardware and Software Configurations

The entire list of hardware and software requirements required for accomplishing the project is covered in this section.

2.1 Hardware Configuration

Hardware	Specifications
Operating System	Windows 10 (64bit)
Processor	Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz
RAM	16 GB
Hard Disk	500 GB
External Hard Disk	1 TB

Figure 1: Hardware configuration

Figure 1 shows a hardware configuration of the system. Figure 4 shows the Windows specification of the system.

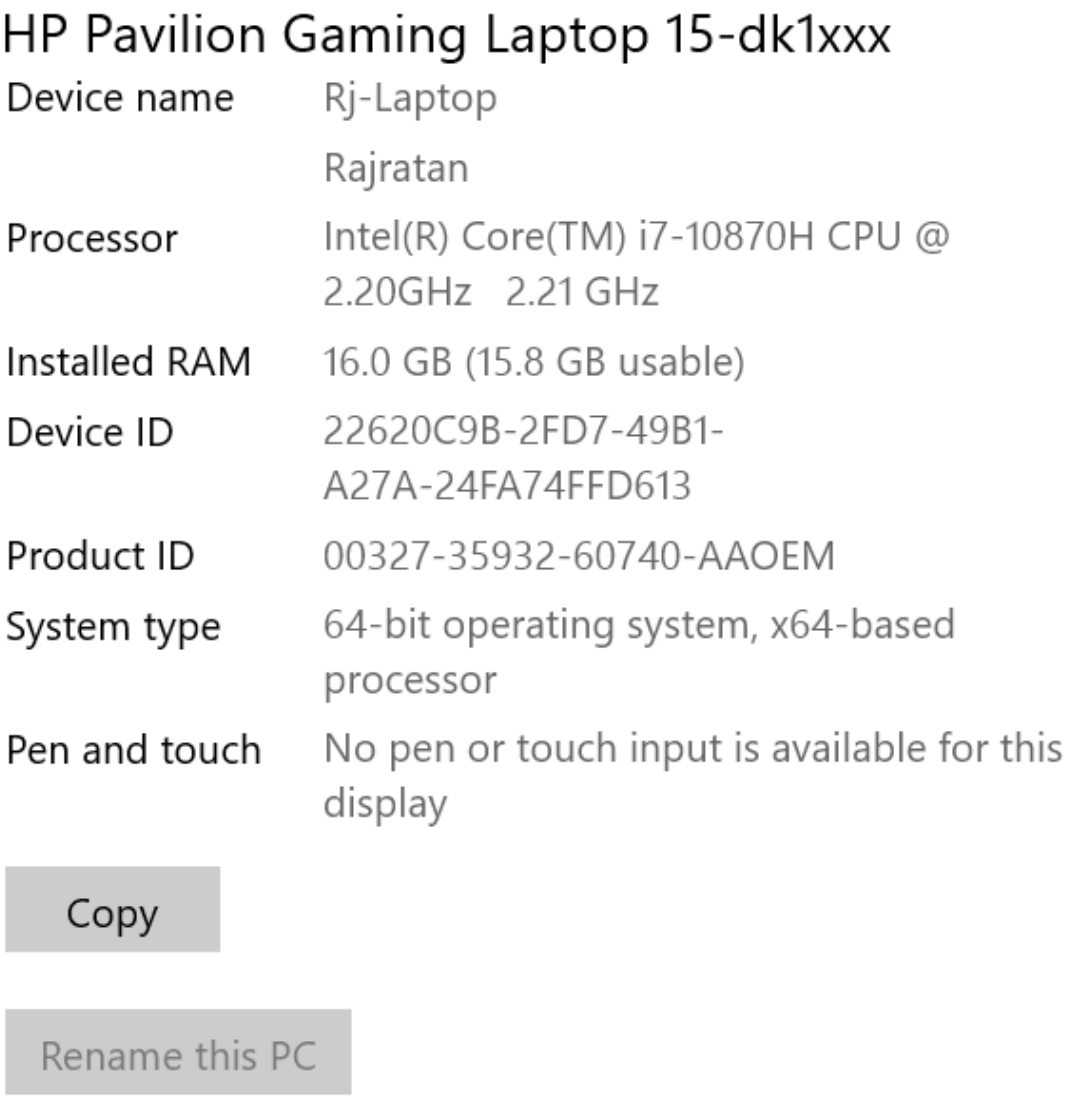

Figure 2: Windows specification

2.2 Software Configuration

Software/Library	Version
Python	3.9
Tensorflow	2.12.0
Matplotlib	3.5.2
OpenCV (cv2)	1.23.5
numpy	4.7.0.72

Figure 3: Software Configuration

Figure 4 shows the Software Configuration of the system.

2.3 Software

2.3.1 PyCharm

PyCharm is a potent integrated development environment (IDE) for Python that comes with databases and web development tools, a strong debugger, comprehensive testing, and version control support. It is cross-platform offers an easy-to-use user experience for effective Python development, and is available in Professional and Community editions.

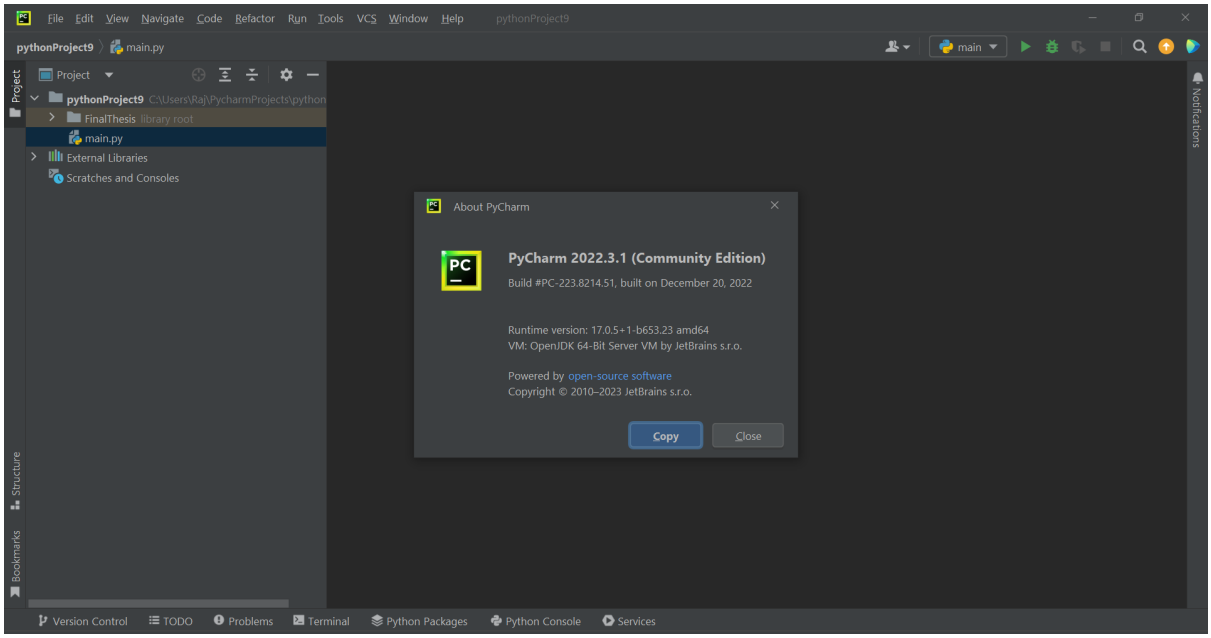

Figure 4: PyCharm software

3 Modelling

In the process of modeling, three distinct convolution neural networks are trained. There are a few standard procedures for all training, as indicated below.

3.1 Proposed Custom CNN

- **Import Necessary Libraries:**

```
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense, BatchNormalization
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

Figure 5: Proposed Custom CNN Required Libraries

- **Data Preprocessing:**

```
train_dataGenerator = ImageDataGenerator(
        rotation_range = 5,
        width_shift_range = 0.1,
        height_shift_range = 0.1,
        rescale = 1.0/255,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = False,
        fill_mode = 'nearest')

test_datagenerator = ImageDataGenerator(rescale=1./255)

train_generator = train_dataGenerator.flow_from_directory(
            "DevanagariHandwrittenCharacterDataset/Train",
            target_size = (32,32),
            batch_size = 32,
            color_mode = "grayscale",
            class_mode = "categorical")


testing_generator = test_datagenerator.flow_from_directory(
            "DevanagariHandwrittenCharacterDataset/Test",
            target_size=(32,32),
            batch_size=32,
            color_mode = "grayscale",
            class_mode='categorical')
```

Figure 6: Data Preprocessing for Custom CNN

- **Build the CNN Model:**

```
cnn_model = Sequential()

cnn_model.add(Conv2D(32, (3,3), strides=(1,1),activation="relu",input_shape=(32,32,1)))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D((2, 2), strides=(2, 2),padding="same"))

cnn_model.add(Conv2D(64, (3,3), strides=(1,1),activation="relu"))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D((2, 2), strides=(2, 2),padding="same"))

cnn_model.add(Conv2D(128, (3,3), strides=(1,1),activation="relu"))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D((2, 2), strides=(2, 2),padding="same"))

cnn_model.add(Flatten())

cnn_model.add(Dense(128, activation="relu", kernel_initializer="uniform"))
cnn_model.add(BatchNormalization())
cnn_model.add(Dropout(0.5))

cnn_model.add(Dense(64, activation="relu", kernel_initializer="uniform"))
cnn_model.add(BatchNormalization())
cnn_model.add(Dropout(0.5))

cnn_model.add(Dense(46, activation="softmax", kernel_initializer="uniform"))
```

Figure 7: Custom CNN Model Structure

- **Compile the Model:**

```
cnn_model.compile(optimizer=Adam(lr=0.0001), loss="categorical_crossentropy", metrics=["accuracy"])
```

Figure 8: Custom CNN Model Compilation

- **Training the Model:**

```
result = cnn_model.fit(train_generator, epochs=20, validation_data=testing_generator)
```

Figure 9: Training of the Custom CNN Model

- **Save the Model:**

```
cnn_model.save("HindiCharacterRecognitionCNNModel.h5")
```

Figure 10: Save Custom CNN Model

3.2 InceptionV3

- **Import Necessary Libraries**

```
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import  Activation, Dropout, Flatten, Dense, BatchNormalization
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

Figure 11: InceptionV3 Required Libraries

- **Data Preprocessing**

```
train_dataGenerator = ImageDataGenerator(
        rotation_range = 5,
        width_shift_range = 0.1,
        height_shift_range = 0.1,
        rescale = 1.0/255,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = False,
        fill_mode = 'nearest')

test_datagenerator = ImageDataGenerator(rescale=1./255)

train_generator = train_dataGenerator.flow_from_directory(
            "DevanagariHandwrittenCharacterDataset/Train",
            target_size = (75,75),
            batch_size = 64,
            color_mode = "rgb",
            class_mode = "categorical")


testing_generator = test_datagenerator.flow_from_directory(
            "DevanagariHandwrittenCharacterDataset/Test",
            target_size=(75,75),
            batch_size=64,
            color_mode = "rgb",
            class_mode='categorical')
```

Figure 12: Data Preprocessing for InceptionV3 Model

- **Build the InceptionV3 Model**

```
pretrained_model = InceptionV3(input_shape=(75, 75, 3), include_top=False, weights='imagenet')

for layer in pretrained_model.layers:
    layer.trainable = False

inception_model = Sequential()
inception_model.add(pretrained_model)
inception_model.add(Flatten())
inception_model.add(Dense(512, activation='relu'))
inception_model.add(BatchNormalization())
inception_model.add(Dropout(0.5))

inception_model.add(Dense(train_generator.num_classes, activation='softmax'))
```

Figure 13: Build the InceptionV3 Model

- **Compile the Model**

```
inception_model.compile(optimizer=Adam(lr=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 14: Compile the InceptionV3 Model

- **Training the Model**

```
result = inception_model.fit(train_generator, epochs=20, validation_data=testing_generator)
```

Figure 15: Training the InceptionV3 Model

- **Save the Model**

```
inception_model.save("HindiCharacterRecognitionInseprtionV3Model.h5")
```

Figure 16: Save the InceptionV3 Model

3.3 Xception

- **Import Necessary Libraries**

```
from tensorflow.keras.applications import Xception
from tensorflow.keras import models
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense, BatchNormalization
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

Figure 17: Xception Required Libraries

- **Data Preprocessing**

```
train_dataGenerator = ImageDataGenerator(
        rotation_range = 5,
        width_shift_range = 0.1,
        height_shift_range = 0.1,
        rescale = 1.0/255,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = False,
        fill_mode = 'nearest')

test_datagenerator = ImageDataGenerator(rescale=1./255)

train_generator = train_dataGenerator.flow_from_directory(
            "DevanagariHandwrittenCharacterDataset/Train",
            target_size = (71,71),
            batch_size = 64,
            color_mode = "rgb",
            class_mode = "categorical")


testing_generator = test_datagenerator.flow_from_directory(
            "DevanagariHandwrittenCharacterDataset/Test",
            target_size=(71,71),
            batch_size=64,
            color_mode = "rgb",
            class_mode='categorical')
```

Figure 18: Data Preprocessing for Xception Model

- **Build the Xception Model**

```
pretrained_model = Xception(include_top=False, weights='imagenet', input_shape=(71, 71, 3))


for layer in pretrained_model.layers:
    layer.trainable = False


inception_model = models.Sequential()
inception_model.add(pretrained_model)
inception_model.add(Flatten())
inception_model.add(Dense(512, activation='relu'))
inception_model.add(BatchNormalization())
inception_model.add(Dropout(0.5))

inception_model.add(Dense(train_generator.num_classes, activation='softmax'))
```

Figure 19: Build the Xception Model

- **Compile the Model**

```
inception_model.compile(optimizer=Adam(lr=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 20: Compile Xception Model

- **Training the Model**

```
result = inception_model.fit(train_generator, epochs=15, validation_data=testing_generator)
```

Figure 21: Training the Xception Model

- **Save the Model**

```
inception_model.save("HindiCharacterRecognitionXceptionModel.h5")
```

Figure 22: Save the Xception Model

3.4 Graphical Presentation of Results

```
plt.plot(epochs_length, accuracy, 'r', label='Training Accuracy')
plt.plot(epochs_length, test_acc, 'g', label='Testing Accuracy')
plt.title('Training and Testing Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.xticks(range(1, len(accuracy), 3))
plt.grid(True)
plt.legend()
plt.figure()


plt.plot(epochs_length, loss, 'r', label='Training Loss')
plt.plot(epochs_length, test_loss, 'g', label='Testing Loss')
plt.title('Training and Testing loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.xticks(range(1, len(loss), 3))
plt.grid(True)
plt.legend()
plt.show()
```

Figure 23: Visual Presentation of custom CNN, InceptionV3,xception Model Results

4 Testing custom CNN model

4.1 Libraries required for testing

```
from tensorflow.keras.preprocessing.image import img_to_array
import cv2
import numpy as np
import tensorflow as tf
```

Figure 24: Libraries required for testing the trained model

4.2 Preprocessing of image

```
testImage = cv2.imread("TestData/kha1.PNG")
curr_image = cv2.resize(testImage, (32,32))
curr_image = curr_image.astype("float") / 255.0
curr_image = img_to_array(curr_image)
curr_image = cv2.cvtColor(curr_image, cv2.COLOR_BGR2GRAY)
curr_image = np.expand_dims(curr_image, axis=0)
curr_image = np.expand_dims(curr_image, axis=3)
```

Figure 25: Preprocessing of image before prediction

4.3 Load trained model

```
Proposed_trained_model = tf.keras.models.load_model("HindiCharacterRecognitionCNNModel.h5")
```

Figure 26: Load trained model

4.4 Predict the result

```
lists_label = Proposed_trained_model.predict(curr_image)[0]
```

Figure 27: Predict the result