National College of Ireland

Configuration Manual

MSc Research Project
Artificial Intelligence

Vineeth Palla
Student ID: x22153187

School of Computing
National College of Ireland

Supervisor: Muslim Jameel Syed

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vineeth Palla		
Student ID:	X22153187		
Programme:	MSc in Artificial Intelligence	**Year:**	2023
Module:	MSc Research Patricum		
Lecturer:	Muslim Jameel Syed		
Submission Due Date:	14/12/2023		
Project Title:	SIGN LANGUAGE RECOGNITIONANDTEXT-TO-SPEECH TRANSLATION		
Word Count :	466	**Page Count :** 8	

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: Vineeth Palla

Date: 14/12/2023

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

Vineeth Palla
X22153187

1 Introduction

A document that aids in our comprehension of the procedures used for this project is the setup manual. It contains instructions for creating, putting into practice, installing, and deploying the "Sign Language Recognition and Text-to-Speech Translation" project that is the subject of this report. This manual's primary goal is to assist and support the user at every step of the process in order to produce the final product and results that are included in this report. All of the details on the software, hardware, and processes needed to carry out this project are contained in the handbook.

2 HARDWARE CONFIGURATION

The specification of the system is as follows,

- Operating system: Windows >= 7
- Processor: Intel i5
- System Compatibility: 64-bit
- Hard Disk: 500GBs
- RAM: >= 8 GB

3 SOFTWARE CONFIGURATIONS

a) Python==3.6.6:-

High-level, interpreted programming is done with Python. Its dynamic typing and dynamic binding, along with its high-level built-in data structures, make it an appealing language for Rapid Application Development and for usage as a scripting or glue language to join existing components. Its object-oriented methodology and language concept are designed to assist programmers in writing logical, understandable code for both small and large-scale projects.

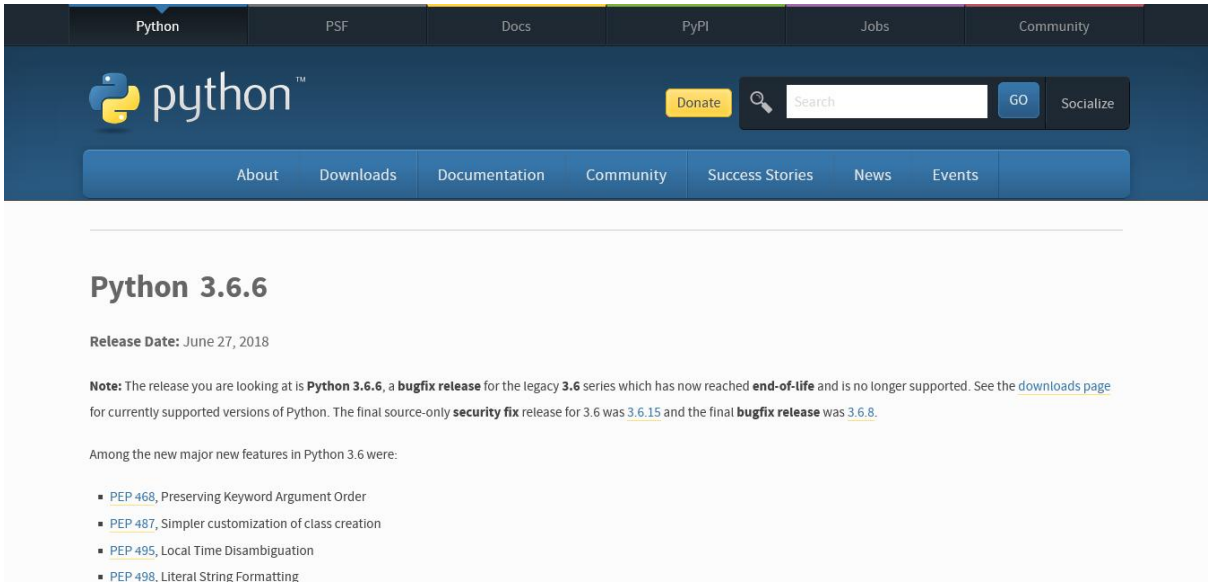

Figure 1: Python Installer.

b) Visual Studio Code:- Microsoft offers a free open source text editor called Visual Studio Code, also referred to as VS Code. Linux, macOS, and Windows can all run VS Code.

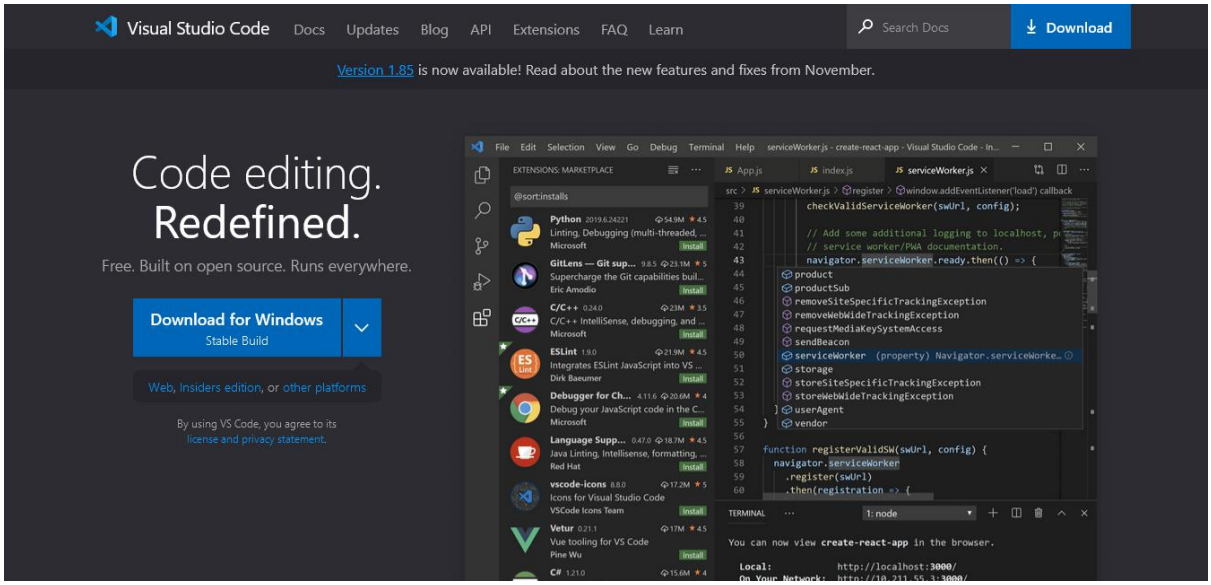

Figure 2: Visual Studio.

4 Libraries Configuration

I. Pandas: A powerful data manipulation and analysis library used for handling structured data.

II. NumPy: Fundamental package for scientific computing with Python, mainly used for numerical operations and working with arrays.

III. Matplotlib: A popular plotting library in Python for creating static, interactive, and publication-quality visualizations.

IV. Keras: A high-level neural network API that is simple to use and is frequently used for quick deep learning experiments.

V. TensorFlow: A Google-created open-source machine learning framework that is frequently used to create and train machine learning models, especially neural networks.

VI. Scikit-learn:A straightforward and effective data mining and analysis tool with a range of machine learning tools and algorithms.

VII. OpenCV (opencv-python): a library of open-source tools for machine learning and computer vision. It offers features and tools for object detection, picture and video analysis, and other uses.

5 Procedure For Machine Learning

5.1 Pre-Processing the data

- Loading the dataset and removing the unwanted columns, duplicate rows, null values and removing other attack categories.

```
def get_image_size():
    img = cv2.imread('gestures/0/100.jpg', 0)
    return img.shape

def get_num_of_classes():
    return len(os.listdir('gestures/'))

image_x, image_y = get_image_size()

def cnn_model():
    num_of_classes = get_num_of_classes()
    model = Sequential()
    model.add(Conv2D(32, (5,5), input_shape=(image_x, image_y, 1), activation='sigmoid'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same'))
    model.add(Conv2D(64, (5,5), activation='sigmoid'))
    model.add(MaxPooling2D(pool_size=(5, 5), strides=(5, 5), padding='same'))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.6))
    model.add(Dense(num_of_classes, activation='softmax'))
    rms_prop = optimizers.RMSprop()
    model.compile(loss='categorical_crossentropy', optimizer=rms_prop, metrics=['accuracy'])
    filepath="new_cnn_model_rmsprop.h5"
    checkpoint1 = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode='max')
    #checkpoint2 = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min'
    callbacks_list = [checkpoint1]
    return model, callbacks_list
```

Figure 3 : Loading thet dataset.

5.2 Training and Testing of models

```
    filepath="new_cnn_model_adam_24sep2022.h5"
    checkpoint1 = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode='max')
    #checkpoint2 = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min'
    callbacks_list = [checkpoint1]
    return model, callbacks_list

def train():
    with open("train_images","r+b") as f:
        train_images = np.array(pickle.load(f))
    with open("train_labels", "r+b") as f:
        train_labels = np.array(pickle.load(f), dtype=np.int32)

    with open("test_images", "r+b") as f:
        test_images = np.array(pickle.load(f))
    with open("test_labels", "r+b") as f:
        test_labels = np.array(pickle.load(f), dtype=np.int32)

    train_images = np.reshape(train_images, (train_images.shape[0], image_x, image_y, 1))
    test_images = np.reshape(test_images, (test_images.shape[0], image_x, image_y, 1))
    train_labels = np_utils.to_categorical(train_labels)
    test_labels = np_utils.to_categorical(test_labels)

    model, callbacks_list = cnn_model()
    model.summary()
    history = model.fit(train_images, train_labels, validation_data=(test_images, test_labels), epochs=5, t
    scores = model.evaluate(test_images, test_labels, verbose=0)
    print("Accuracy: ", scores[1])
    #accuracy and loss plot
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('Accuracy')
    plt.ylabel('accuracy')
```

Figure 4 : Training and testing of Gestures dataset and using Adam's Model.

```
def train():
    with open("train_images","r+b") as f:
        train_images = np.array(pickle.load(f))
    with open("train_labels", "r+b") as f:
        train_labels = np.array(pickle.load(f), dtype=np.int32)

    with open("test_images", "r+b") as f:
        test_images = np.array(pickle.load(f))
    with open("test_labels", "r+b") as f:
        test_labels = np.array(pickle.load(f), dtype=np.int32)

    train_images = np.reshape(train_images, (train_images.shape[0], image_x, image_y, 1))
    test_images = np.reshape(test_images, (test_images.shape[0], image_x, image_y, 1))
    train_labels = np_utils.to_categorical(train_labels)
    test_labels = np_utils.to_categorical(test_labels)

    model, callbacks_list = cnn_model()
    model.summary()
    history = model.fit(train_images, train_labels, validation_data=(test_images, test_labels), epochs=5, t
    scores = model.evaluate(test_images, test_labels, verbose=0)
    print("Accuracy: ", scores[1])
    #accuracy and loss plot
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('Accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train','val'], loc='upper left')
    plt.show()
```

Figure 5 : Training and testing of Gestures dataset and using RMSProps's Model.

```python
def train():
    with open("train_images","r+b") as f:
        train_images = np.array(pickle.load(f))
    with open("train_labels", "r+b") as f:
        train_labels = np.array(pickle.load(f), dtype=np.int32)

    with open("test_images", "r+b") as f:
        test_images = np.array(pickle.load(f))
    with open("test_labels", "r+b") as f:
        test_labels = np.array(pickle.load(f), dtype=np.int32)

    train_images = np.reshape(train_images, (train_images.shape[0], image_x, image_y, 1))
    test_images = np.reshape(test_images, (test_images.shape[0], image_x, image_y, 1))
    train_labels = np_utils.to_categorical(train_labels)
    test_labels = np_utils.to_categorical(test_labels)

    model, callbacks_list = cnn_model()
    model.summary()
    history = model.fit(train_images, train_labels, validation_data=(test_images, test_labels), epochs=5,
    scores = model.evaluate(test_images, test_labels, verbose=0)
    print("Accuracy: ", scores[1])
    #accuracy and loss plot
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('Accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train','val'], loc='upper left')
    plt.show()

    #loss plot
    plt.plot(history.history['loss'])
```

Figure 6 : Training and testing of Gestures dataset and using SGD's Model.

5.3 Results

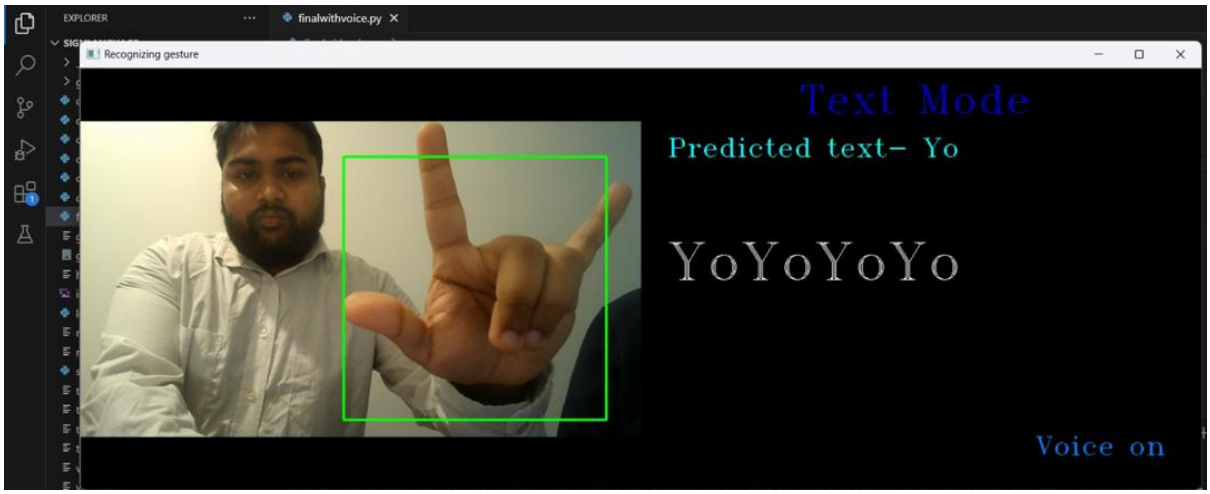

Figure 7 : Results for the following sign Gestures.

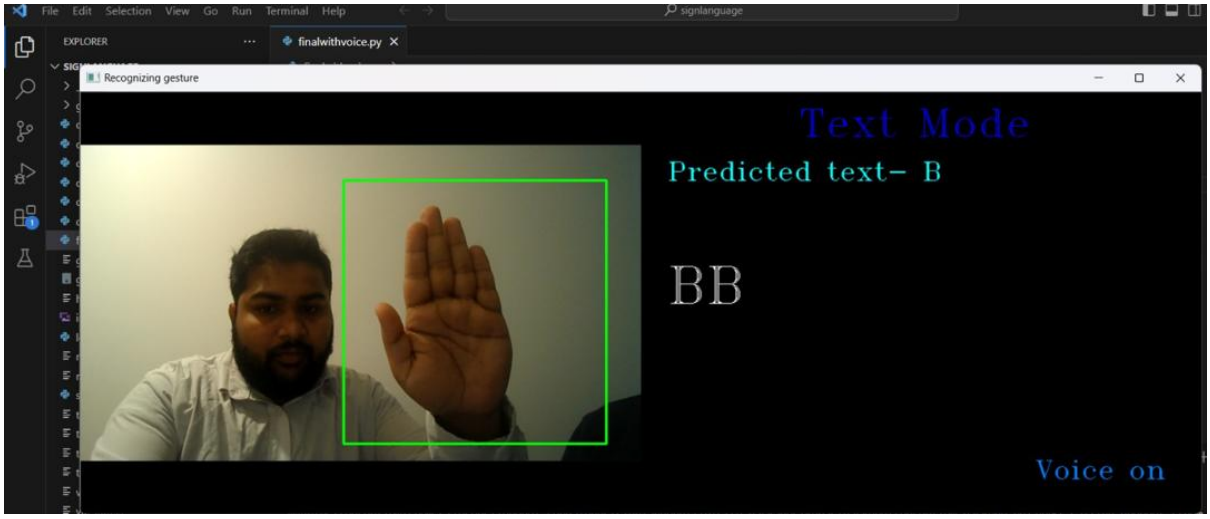

Figure 8 : Results for the following sign Gestures.

6 REFERENCE

Python 3.6.6
Available at: https://www.python.org/downloads/release/python-363/

Visual Studio Code
Available at : https://code.visualstudio.com/