Configuration Manual

MSc Research Project
Data Analytics

Pritish Mehta
Student ID: x20184409

School of Computing
National College of Ireland

Supervisor: Giovani Estrada

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Pritish Mehta
Student ID:	x20184409
Programme:	MSc in Data Analytics
Module:	Research Project
Supervisor:	Giovani Estrada
Submission Due Date:	15/08/2022
Project Title:	A Novel Combination Of 3D CNN's And Recurrent Neural Networks for Sign Language to Text Conversion
Word Count:	700 **Page Count:** 4

Programme: MSc in Data Analytics

Signature: Pritish Mehta

Date: 15/08/2022

Configuration Manual

Pritish Mehta
Student ID: x20184409

1 Introduction

This setup manual includes all of the information required to run the artifact. It offers an overview of the thesis work's minimal and suggested prerequisites for replication. This manual bridges the gap between the artifact and the thesis. Along with the software and hardware requirements for this thesis, all of the essential components of this thesis are detailed below using code snippets. It includes directions for gathering data, running the artifact, and presenting the artifact's noteworthy results.

2 Hardware Requirements

Processor: AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
Memory (RAM) Installed: 16 GB DDR4 3200 MHz
System Type: Windows 10 Pro, 64 Bit
Operating System with x64-based processor
Storage: 500 GB SSD and 1TB HDD
GPU: 12 GB, Nvidia GeForce GTX 1650

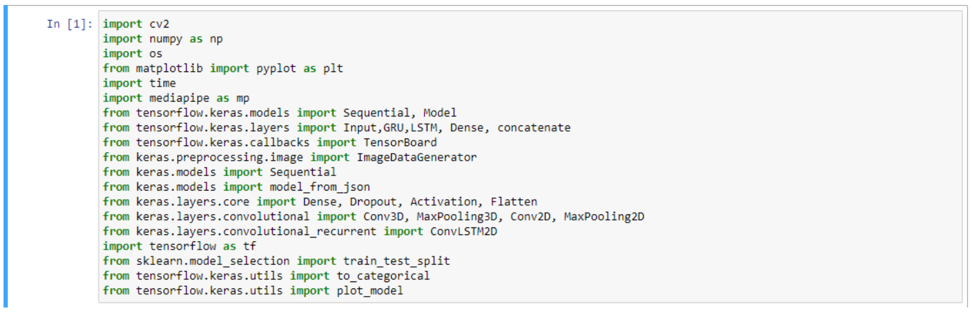
3 Software Requirements

Jupyter Notebook was utilized as the Integrated Development Environment (IDE) for this project, while Python was used as the programming language. The visualizations were created with the help of the seaborn and matplotlib packages. The precise versions of these programs are listed below.

Python 3.9.12
Jupyter Notebook: 6.4.8

Python 3.9.12 is utilized for all coding sections throughout the study. Anaconda Navigator platform is installed, which includes Jupyter Lab, Jupyter Notebook, and the ability to open a Python 3 file to launch and execute code. Anaconda's 64-bit version for Windows 10 must be installed. After a successful installation, run Anaconda Navigator (Fig. 3), then launch Jupyter lab or Jupyter notebook. When we click launch, it will automatically open in Brave Browser.

4 Library Package Requirements



```
In [1]: import cv2
        import numpy as np
        import os
        from matplotlib import pyplot as plt
        import time
        import mediapipe as mp
        from tensorflow.keras.models import Sequential, Model
        from tensorflow.keras.layers import Input,GRU,LSTM, Dense, concatenate
        from tensorflow.keras.callbacks import TensorBoard
        from keras.preprocessing.image import ImageDataGenerator
        from keras.models import Sequential
        from keras.models import model_from_json
        from keras.layers.core import Dense, Dropout, Activation, Flatten
        from keras.layers.convolutional import Conv3D, MaxPooling3D, Conv2D, MaxPooling2D
        from keras.layers.convolutional_recurrent import ConvLSTM2D
        import tensorflow as tf
        from sklearn.model_selection import train_test_split
        from tensorflow.keras.utils import to_categorical
        from tensorflow.keras.utils import plot_model
```

Figure 1: Packages Used

The required python packages installed in the environment is listed below. ’pip’ command is used to install all packages.

- Keras 1.0.8
- Tensorflow 2.5.0
- Numpy 1.21.5
- Matplotlib 3.5.1
- Open CV 4.6.0

5 Dataset Description

The dataset is extracted from WLASL website which links to their own public repository. There is a python file along with a csv file containing the video glosses. The Python code is used to download the 28,000 videos. The videos are stored in videos folder.

Link to the WLASL dataset

https://dxli94.github.io/WLASL/

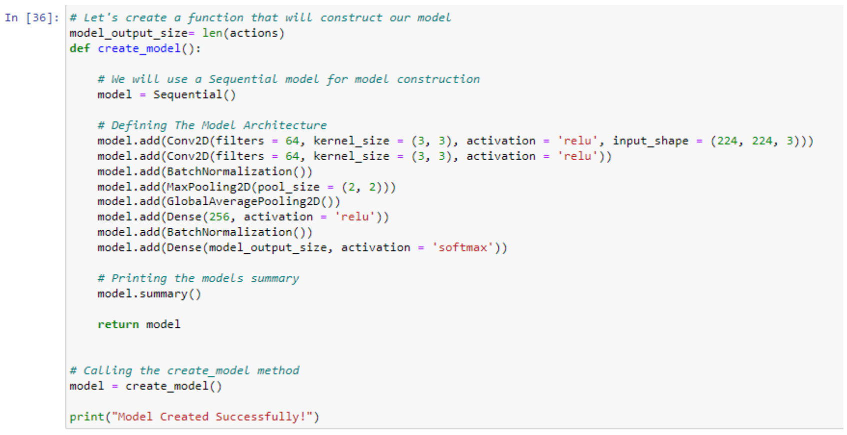
6 Dataset Preparation and Pre-processing

Once the dataset is downloaded to create the frames and the numpy arrays that are necessary to run the models run all lines in “Creating_frames_and_npy_files.ipynb” file which will automatically create frames and npy files of all the videos in the “videos” folder.

7 Model Preparation

In the folder run the model which is required the are all labelled "ASL (Model_Name).ipynb". Run all the line in the file to run the model.

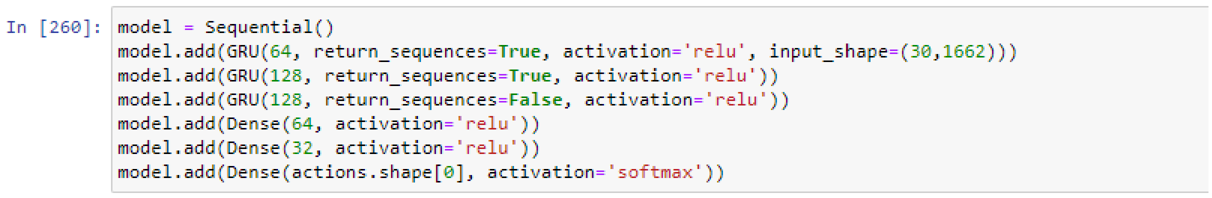
7.1 CNN Model



```
In [36]: # Let's create a function that will construct our model
         model_output_size= len(actions)
         def create_model():

             # We will use a Sequential model for model construction
             model = Sequential()

             # Defining The Model Architecture
             model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', input_shape = (224, 224, 3)))
             model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
             model.add(BatchNormalization())
             model.add(MaxPooling2D(pool_size = (2, 2)))
             model.add(GlobalAveragePooling2D())
             model.add(Dense(256, activation = 'relu'))
             model.add(BatchNormalization())
             model.add(Dense(model_output_size, activation = 'softmax'))

             # Printing the models summary
             model.summary()

             return model


         # Calling the create_model method
         model = create_model()

         print("Model Created Successfully!")
```

Figure 2: CNN Model Process Flow

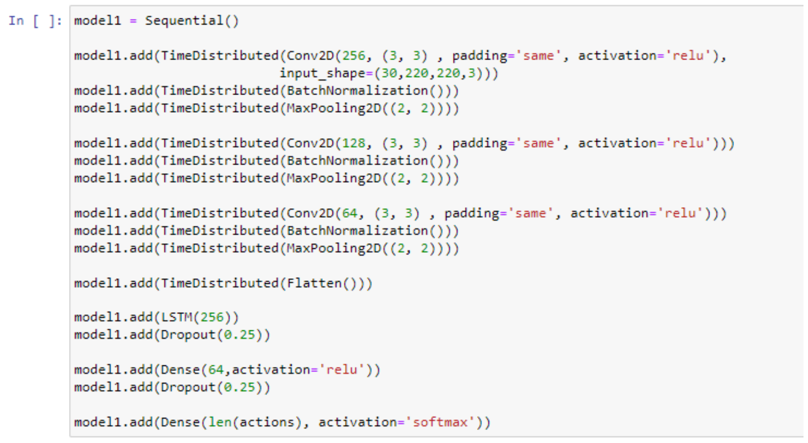
7.2 GRU Model



```
In [260]: model = Sequential()
          model.add(GRU(64, return_sequences=True, activation='relu', input_shape=(30,1662)))
          model.add(GRU(128, return_sequences=True, activation='relu'))
          model.add(GRU(128, return_sequences=False, activation='relu'))
          model.add(Dense(64, activation='relu'))
          model.add(Dense(32, activation='relu'))
          model.add(Dense(actions.shape[0], activation='softmax'))
```

Figure 3: GRU Model Process Flow

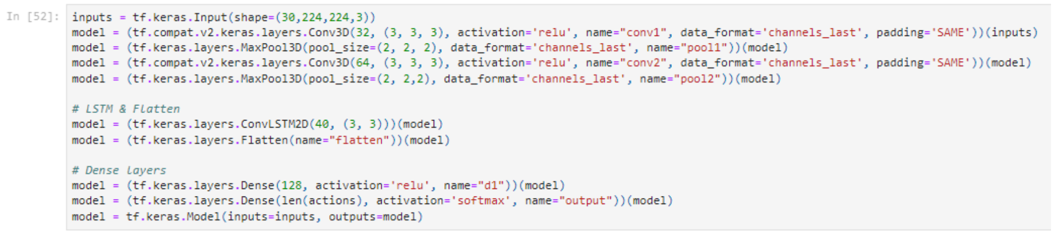
7.3 CNN + LSTM Model



```
In [ ]: model1 = Sequential()

        model1.add(TimeDistributed(Conv2D(256, (3, 3) , padding='same', activation='relu'),
                                   input_shape=(30,220,220,3)))
        model1.add(TimeDistributed(BatchNormalization()))
        model1.add(TimeDistributed(MaxPooling2D((2, 2))))

        model1.add(TimeDistributed(Conv2D(128, (3, 3) , padding='same', activation='relu')))
        model1.add(TimeDistributed(BatchNormalization()))
        model1.add(TimeDistributed(MaxPooling2D((2, 2))))

        model1.add(TimeDistributed(Conv2D(64, (3, 3) , padding='same', activation='relu')))
        model1.add(TimeDistributed(BatchNormalization()))
        model1.add(TimeDistributed(MaxPooling2D((2, 2))))

        model1.add(TimeDistributed(Flatten()))

        model1.add(LSTM(256))
        model1.add(Dropout(0.25))

        model1.add(Dense(64,activation='relu'))
        model1.add(Dropout(0.25))

        model1.add(Dense(len(actions), activation='softmax'))
```

Figure 4: CNN+LSTM Model Process Flow

7.4 3D-CNN + LSTM Model



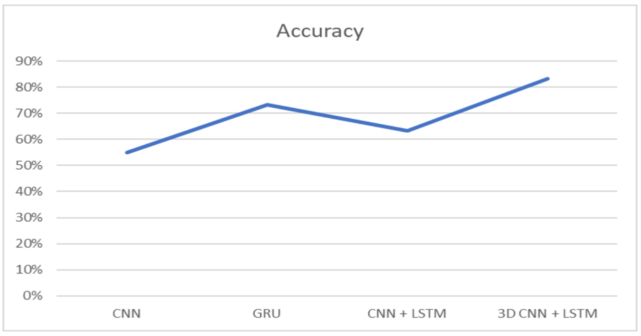
```
In [52]: inputs = tf.keras.Input(shape=(30,224,224,3))
         model = (tf.compat.v2.keras.layers.Conv3D(32, (3, 3, 3), activation='relu', name="conv1", data_format='channels_last', padding='SAME'))(inputs)
         model = (tf.keras.layers.MaxPool3D(pool_size=(2, 2, 2), data_format='channels_last', name="pool1"))(model)
         model = (tf.compat.v2.keras.layers.Conv3D(64, (3, 3, 3), activation='relu', name="conv2", data_format='channels_last', padding='SAME'))(model)
         model = (tf.keras.layers.MaxPool3D(pool_size=(2, 2,2), data_format='channels_last', name="pool2"))(model)

         # LSTM & Flatten
         model = (tf.keras.layers.ConvLSTM2D(40, (3, 3)))(model)
         model = (tf.keras.layers.Flatten(name="flatten"))(model)

         # Dense Layers
         model = (tf.keras.layers.Dense(128, activation='relu', name="d1"))(model)
         model = (tf.keras.layers.Dense(len(actions), activation='softmax', name="output"))(model)
         model = tf.keras.Model(inputs=inputs, outputs=model)
```

Figure 5: 3d-CNN+LSTM Model Process Flow

8 American Sign Language Translation

The Models that are using in this research has the following parameters.

Model	Epochs	Training Time (s)	Accuracy
CNN	50	2,350	73.33%
GRU	200	1,500	55%
CNN + LSTM	200	18,000	63.33%
3D CNN + LSTM	200	28,800	83.333%



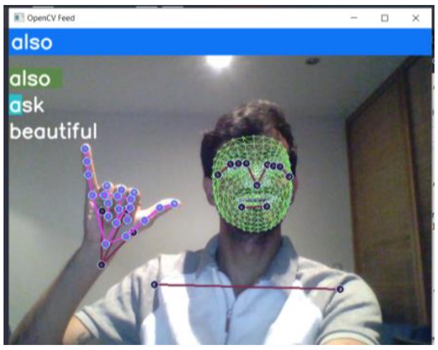
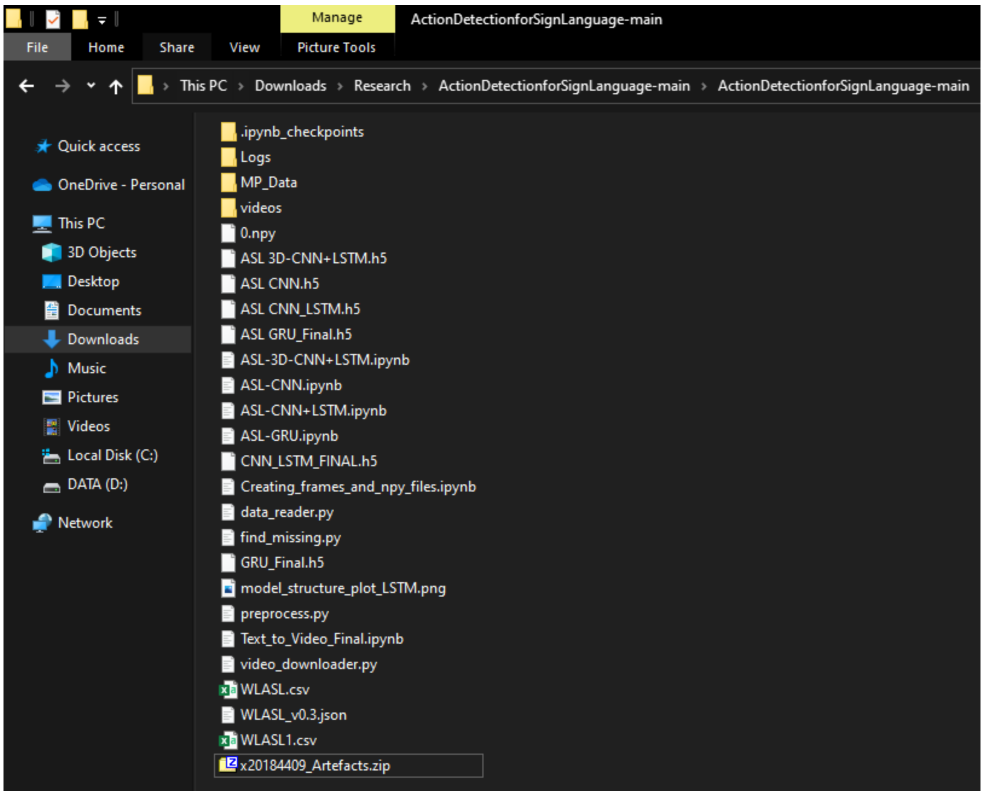


Figure 6: Model Evaluation

8.1 Structure of the Directory

The Structure of the directory will look like the image below



.ipynb_checkpoints
Logs
MP_Data
videos
0.npy
ASL 3D-CNN+LSTM.h5
ASL CNN.h5
ASL CNN_LSTM.h5
ASL GRU_Final.h5
ASL-3D-CNN+LSTM.ipynb
ASL-CNN.ipynb
ASL-CNN+LSTM.ipynb
ASL-GRU.ipynb
CNN_LSTM_FINAL.h5
Creating_frames_and_npy_files.ipynb
data_reader.py
find_missing.py
GRU_Final.h5
model_structure_plot_LSTM.png
preprocess.py
Text_to_Video_Final.ipynb
video_downloader.py
WLASL.csv
WLASL_v0.3.json
WLASL1.csv
x20184409_Artefacts.zip

Figure 7: Structure of the Directory

8.2 Steps to trigger the ASL Recognition and Translation System

The system is divided into 3 parts

1. Data Pre-Processing

    a. Download the dataset from the ".py" file in the folder.
    b. All the videos are stored in the "videos" folder.
    c. Run the "Creating_frames_and_npy_files.ipynb" which will create ".npy" and ".jpg" frames.
    d. After the frames are created "MP_Data" folder will be created containing all the words and the corresponding "npy" and "jpg" files

2. Model Training

    a. Once the dataset is created next step is to feed the pre-processed data into the model.
    b. Each model has its own ".ipynb" files. Run all the lines of code to run the model.
    c. The Model code is divided into 10 sections where 1-7 is model training and saving the model.

3. Model Evaluation

    a. The final step is to evaluate the model.
    b. For which we will load the trained model and run it directly.
    c. The steps 1-5 needs to be run first then we can run step 8-10 which is loading the model and evaluating.