National College of Ireland

Sign Language detection in Cloud computing environment using Deep learning Algorithms Configuration Manual

MSc Research Project
Cloud Computing

Monika Malik
Student ID: X20149611

School of Computing
National College of Ireland

Supervisor: Mr Aqeel Kazmi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …MONIKA MALIK….

Student ID: ……X20149611……

Programme: ……MSC IN CLOUD COMPUTING …… **Year:** 2021

Module: …..…… RESEARCH PROJECT ……..……

Lecturer: ……Aqeel Kazmi…….……

Submission Due Date: ……16/12/2021..……..……

Project Title: Sign Language detection in Cloud computing environment using Deep learning Algorithms

……..……

Word Count: …… **Page Count:** ……..……..……

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: ……Monika Malik……

Date: ……15/12/2021……

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

Monika Malik
Student ID: X20149611

1 Overview

This research work is related to Sign Language detection in which gesture is recognized by using Deep learning Algorithms in cloud computing environment. The sensor gloves and camera-based system are the two primary strategies which have been used for developed the contact-based system. When we communicate through sign language, then we have use the hand movements and the facial expressions. The letters of the American sign language (ASL) and the Indian sign language (ISL) they made with the hands and they do not require the expression of emotions on the face.This study utilize Deep Learning Model such as VGG-19, Resnet-50, and custom build models, which programmatical implementation will be discussed further

2 Prerequisite

The section describes the required python packages and libraries that need to run the project. The figure 1 shows all required packages.

2.1 Python

In this project we are using python to simulate its working scenario, so it is important to be installed. Currently, there are two versions of python available python 2 and python 3. To run this project, we need python 3 to be installed.

2.2 NumPy

The numpy package is used to conduct the Weight mean average (WMA). Numpy is the scientific computing toolkit which is written in Python that is also used in the Panda for the data analysis.

2.3 Sklearn

Sklearn is generally python's most useful machine learning library. One of the purpose of importing sklearn library is to split the data in train and test and it is also used to encode the categorial value into the numeric values. It includes the classification, regression, clustering and dimensionally reduction are the useful capabilities in the sklearn toolkit for machine learning and the statistical modelling.

2.4 Pickle

Pickle is used in python object structure and is basically used for store the data. It is used to transform a python object into a byte flow in order to save it to a file of the database, in order to retain the state of the application across all send data over the network sessions.

2.5 Matplotlib

Matplotlib is a graphical plotting toolkit. Python and its numerical extension NumPy are both cross-platform and run on all platforms. It is an open source replacement for MATLAB.

2.6 Tensor flow

Tensor flow is an open and free source library for the numerical computing. Google introduced and maintains it, and it is licensed under the Apache 2.0 open-source license. This same API is apparently for the Python programming language, but it also provides access to the entire C++ API. It is capable of running on single CPU systems, GPUs, mobile devices, and on the large-scale distributed systems with the hundreds of machines.

```
import cv2
import os
import pandas as pd
import numpy as np
from tqdm import tqdm
import pickle
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
```

```
from tensorflow.python.keras import utils
from tensorflow_addons.metrics import F1Score
from tensorflow.python.keras.applications import vgg19,resnet
from tensorflow.python.keras.models import Sequential,load_model
from tensorflow.python.keras.layers import Conv2D,Flatten,BatchNormalization,Dense,Dropout,Input,MaxPooling2D
from sklearn.preprocessing import OneHotEncoder
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras.metrics import Precision,Recall
```

Figure 1. Importing packages

3 Data preparation for Experiments

This section will explain how to upload data in google drive that is connected with Colab. Firstly, unzip the *artefact.zip* file, the structure of the files should be similar as shown in fig 2. The tree like structure shown in figure is built by (Friend, N. 2020)

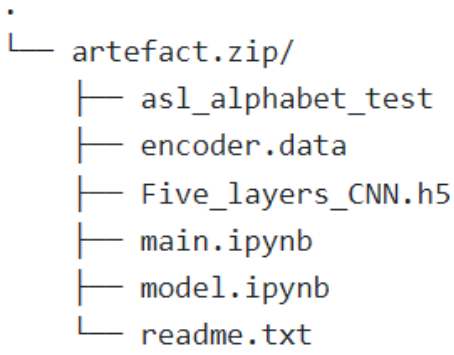

Figure 2. Files directory

asl_alphabet_test: This folder contains samples of sign language images for testing purpose.

Encoder .data :- The encoder .data is define as the binary fie of actual labels of training data.

Five layers CNN .h5 :- This is the best saved file of best model on the basics of ealuation metrics.

Main .ipynb :- This is the main jupyter notebook.

Model .ipynb :- This is the jupyter notebook for model training.

Readme .txt :- Readme file contains dataset link.

3 Google Colaboratory Environment Setup

Google Colab is used to execute the project. In our dataset, ASL Alphabet has the large number of pictures that's why we are using colab a free cloud service for jupyter notebook file execution. Jupyter notebooks can be run freely in Google Colab. The colab environment already has all of the necessary packages and libraries installed. The free edition of Colab comes with 12 GB of RAM and a remote hard disk of 38 GB.

1. Go to https://colab.research.google.com/ and login in with any Gmail account to use the Colab.
2. Three types of run types are available in google colab i.e. None, GPU, and TPU. We'll select none as shown in fig.3

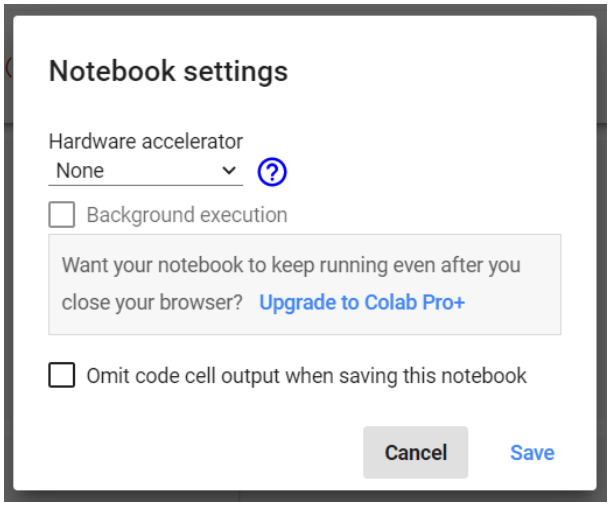

Figure 3. Colab Run-type

3. Now mount your google drive in Colab notebook as shown in fig. 5. The path directory needs to be change if required. Upload the code and data file to this directory. The details of the files are explained in section 2.

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

[ ] import os
    os.chdir('/content/drive/My Drive/thesis/monika_final_code (2)')
    !pwd

    /content/drive/My Drive/thesis/monika_final_code (2)
```

Fig.4 Drive Mount.

IMPLEMENTATION

There are 29 classes and batch size is 4 and the epoche size is 10.
Set the system parameters and variables with appropriate values after mounting the Google drive in Colab and importing the required packages and files (fig.4)

```
train_dir = r".\asl_alphabet_train\asl_alphabet_train"
test_dir = r".\asl_alphabet_test\asl_alphabet_test"

classes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
           'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
           'W', 'X', 'Y', 'Z', 'nothing', 'space', 'del']
```

```
img_target_size = 64,64

classes_len = 29
batch_size = 4
epochs_size = 10
```

Fig. 5 System parameters

```
train_gen = ImageDataGenerator(rotation_range = 40,
                               shear_range = 0.2,
                               zoom_range = 0.2,
                               rescale=1./255.0,
                               horizontal_flip = True,
                               validation_split=0.3)
val_gen = ImageDataGenerator(rescale=1./255.0, validation_split=0.3)


train_data = train_gen.flow_from_directory(directory=r"asl_alphabet_train\asl_alphabet_train",
                           target_size=img_target_size,
                           batch_size=batch_size,
                           subset="training"
                          )
val_data = val_gen.flow_from_directory(directory=r"asl_alphabet_train\asl_alphabet_train",
                           target_size=img_target_size,
                           batch_size=batch_size,
                           subset="validation"
                       )
```

```
Found 60900 images belonging to 29 classes.
Found 26100 images belonging to 29 classes.
```

Fig.6 Pre-processing of training data

VGG-19

VGG-19 load pre-trained VGG model with imagenet weights

```
vgg_19 = vgg19.VGG19(include_top=False,weights="imagenet",input_shape=(img_target_size[0],img_target_size[1],3))

model = Sequential()
model.add(vgg_19)
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(Dense(classes_len, activation='softmax'))

model.layers[0].trainable=False
```

Fig. 7 VGG-19

RESNET 50

RESNET 50 load pre-trained RESNET50 model with imagenet weights.

```
resnet = resnet.ResNet50(include_top=False,weights="imagenet",input_shape=(img_target_size[0],img_target_size[1],3))

model = Sequential()
model.add(resnet)
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(Dense(classes_len, activation='softmax'))

model.layers[0].trainable=False
```

```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy",Precision(),Recall(),F1Score(num_classes=cla
sses_len)])
```

Fig. 8 RESNET 50

CUSTOM MODEL WITH THREE LAYER

Custom model with three layer with a three convolution layer having “relu” as a activation function following a dense layer with “sigmoid” activation layer.

```
model = Sequential()

model.add(Conv2D(64, (3, 3), padding='same', input_shape=(img_target_size[0], img_target_size[1], 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(1024, activation='sigmoid'))
model.add(Dense(classes_len, activation='softmax'))
```

```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy",Precision(),Recall(),F1Score(num_classes=cla
sses_len)])
```

Fig. 9 Custom model with three layer

CUSTOM MODEL WITH FIVE LAYER

Custom model with five convolution layer having same “relu” as an activation function but the dense model is having two layer first dense layer with “sigmoid” activation layer and second dense layer with “softmax”.

```
# We are using this model


model = Sequential()

model.add(Conv2D(64, (3, 3), padding='same', input_shape=(img_target_size[0], img_target_size[1], 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Conv2D(256, (3, 3),  padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(512, (3, 3),  padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(1024, activation='sigmoid'))
model.add(Dense(classes_len, activation='softmax'))
```

Fig .10 Custom model with five layer

Save the best model in .h5 extension and dump the actual labels of training dataset using pickle file.

```
[ ]  model.save("Five_layers_CNN.h5")
```

```
[ ]  image = plt.imread(r"/content/drive/My Drive/thesis/monika_final_code (2)/asl_alphabet_test/asl_alphabet_test/D_test.jpg")
     image = cv2.resize(image,(64,64))
     image = image/255.0
     image = image.reshape((-1,64,64,3))
     pred = model.predict(image)
```

```
[ ]  labels = {}
     actual = train_data.class_indices
     for i in train_data.class_indices.keys():
         labels[actual[i]] = i
```

```
[ ]  pickle.dump(labels, open("encoder.data", "wb"))
```

Fig. 11 Save model

For real time prediction via webcam. Open the main.ipynb in local machine using jupyter notebook.
Load the save files in notebook and define a function for alphabet prediction.

```
In [3]: mp_hands = mp.solutions.hands
```

```
In [4]: model = load_model("Five_layers_CNN.h5")
        label_names = pickle.load(open("encoder.data", 'rb'))
```

```
In [5]: def pred_alphabet(cur_image) :

            target_shape = model.input_shape[1:3]
            cur_image = cv2.resize(cur_image, target_shape)
            cur_image = cur_image.astype('float32')/255.0
            cur_image = cur_image.reshape((-1,64,64,3))
            predictions = model.predict(cur_image)
            return label_names[np.argmax(predictions)]
```

Fig. 12 Loading the save files

Real time streaming from file directory.

Stream from Directory

In [6]:
```
test = r"asl_alphabet_test/asl_alphabet_test/"
for img in os.listdir(test):
    image = cv2.imread(test+img)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    with mp_hands.Hands(min_detection_confidence=0.5,min_tracking_confidence=0.5) as hands:
        results = hands.process(image)
        if results.multi_hand_landmarks:
            xList = []
            yList = []
            bbox = []
            myHand = results.multi_hand_landmarks[0]
            x,y=[],[]
            for id, lm in enumerate(myHand.landmark):
                h, w, c = image.shape
                px, py = int(lm.x * w), int(lm.y * h)
                x.append(px)
                y.append(py)
            xmin, xmax = min(x), max(x)
            ymin, ymax = min(y), max(y)
            boxW, boxH = xmax - xmin, ymax - ymin
            bbox = xmin, ymin, boxW, boxH
            w_pad = 50
            h_pad = 100
            y1,y2 = bbox[1]-h_pad ,bbox[1] + bbox[3] +h_pad
            x1,x2 = bbox[0]-w_pad, bbox[0] + bbox[2]+w_pad
            if x1 <0 :
                x1 = 0
            if y1 <0 :
                y1 = 0
            cropped_image = image[y1:y2,x1:x2,:]
            pred_label = pred_alphabet(cropped_image)
            image = cv2.rectangle(image,(x1,y1),(x2,y2),(0, 255, 0), 2)

            font = cv2.FONT_HERSHEY_SIMPLEX
            org = (50, 50)
            fontScale = 1
            color = (255, 0, 0)
            thickness = 2

            image = cv2.putText(image,pred_label,org,font,fontScale, color, thickness, cv2.LINE_AA)

            plt.imshow(image)
            plt.show()
```

Fig. 13 Stream for directory

Real time streaming using OpenCV.

Live Stream

In [7]:
```
cap = cv2.VideoCapture(0)
with mp_hands.Hands(min_detection_confidence=0.5,min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
        image = cap.read()[1]
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = hands.process(image)
        if results.multi_hand_landmarks:
            xList = []
            yList = []
            bbox = []

            myHand = results.multi_hand_landmarks[0]
            x,y=[],[]
            for id, lm in enumerate(myHand.landmark):
                h, w, c = image.shape
                px, py = int(lm.x * w), int(lm.y * h)
                x.append(px)
                y.append(py)
            xmin, xmax = min(x), max(x)
            ymin, ymax = min(y), max(y)
            boxW, boxH = xmax - xmin, ymax - ymin
            bbox = xmin, ymin, boxW, boxH
            w_pad = 50
            h_pad = 100
            y1,y2 = bbox[1]-h_pad ,bbox[1] + bbox[3] +h_pad
            x1,x2 = bbox[0]-w_pad, bbox[0] + bbox[2]+w_pad
            if x1 <0 :
                x1 = 0
            if y1 <0 :
                y1 = 0
            cropped_image = image[y1:y2,x1:x2,:]
            pred_label = pred_alphabet(cropped_image)
            image = cv2.rectangle(image,(x1,y1),(x2,y2),(0, 255, 0), 2)

            font = cv2.FONT_HERSHEY_SIMPLEX
            org = (50, 50)
            fontScale = 1
            color = (255, 0, 0)
            thickness = 2

            image = cv2.putText(image,pred_label,org,font,fontScale, color, thickness, cv2.LINE_AA)
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        cv2.imshow("Image",image)
        if cv2.waitKey(5) & 0xFF == 27:
            break
cap.release()q
```

Fig.13 Live stream

References

Friend, N. (2020) tree-online: An online tree-like utility for generating ASCII folder structure diagrams. Written in TypeScript and React.

Paszke, A. et al., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.