Machine Learning – IT4060

Assignment II

B.Sc. (Hons) in Information Technology Specializing in Software Engineering

Department of Computer Science and Software Engineering
Sri Lanka Institute of Information Technology

Sri Lanka

May 2022

Contents

Introduction

A normal person with a hearing threshold of 20db or greater in both ears is considered as a person without any hearing disabilities, anyone having a lesser hearing threshold is considered as a person with a mild, moderate, severe, or profound hearing loss. Persons ranging from mild to severe hearing impairments are categorized as ‘Hard of Hearing’. They communicate through spoken language and with the help of hearing aids. Persons with profound hearing loss is categorized as ‘Deaf’. Deaf people have extremely weak hearing or rather no hearing at all and they communicate using sign language.

Sign language is a non-verbal communication medium which consists of facial expressions, hand signals, gestures, and body language. According to the World Health Organization over 5% of the world’s population are affected by hearing loss and deafness. This includes approximately 430 million adults and 34 million children. Even though hearing disability is a familiar topic in the society, there is less awareness regarding the difficulties faced by this community.

The main dilemma faced by the hearing-impaired community is the prevailing communication barrier between them and the people who communicate with spoken languages. Majority of the persons without any hearing disabilities lacks sign language knowledge. This situation makes the lives of the hearing-impaired community much more complicated as they are unable to effectively interact with the society to fulfil their needs and wants.

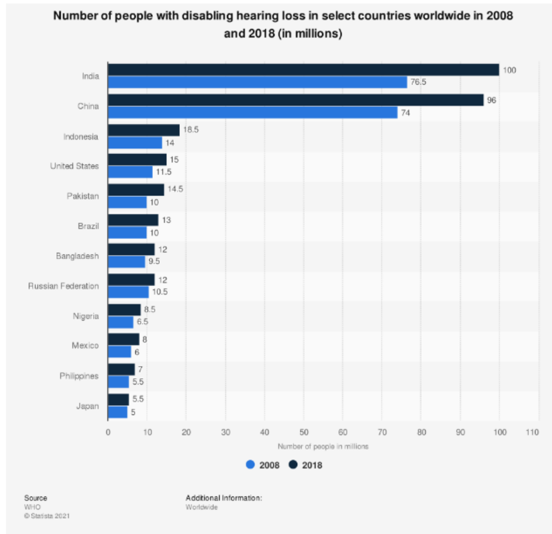

Figure 1 - WHO Statistic of Hearing Loss

The proposed solution is a minor attempt to bridge the prevailing communication barrier between the two communities. We have implemented a model using machine learning techniques which is capable of recognizing sign language gestures that represent letters of the English alphabet. Any sign gesture representing a letter of the English alphabet can be presented in front of a camera and the corresponding letter would appear on the screen of the user. This solution can be used by both communities for communication as well as for educational purposes.

Dataset

The model was trained using a dataset that was downloaded from Kaggle which contains images of the English alphabet relevant to the American sign language. The images have been separated into 29 different folders representing various classes. The dataset has been categorized as ‘Training’ and ‘Testing’ data. The training dataset comprises of images representing the 26 letters of the English alphabet and 3 other gestures representing ‘Delete’, ‘Nothing’ and ‘Space’. Each class comprises of over 7000 elements. The testing dataset consists of 29 images that can be used to test the speed and accuracy of the model.

https://www.kaggle.com/datasets/debashishsau/aslamerican-sign-language-aplhabet-dataset

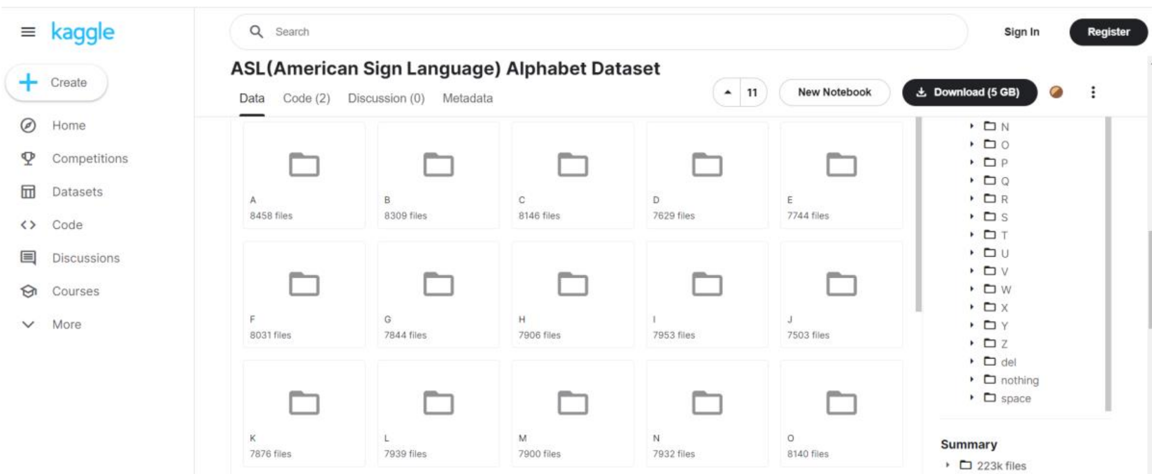

Figure 2 - Kaggle Dataset

Methodology

VGG 16 algorithm was used to train the model. VGG 16 is a convolutional neural network model which was used to win the Imagenet competition in 2016. VGG 16 is considered to be an excellent object detection model. The VGG 16 model consists of convolutional layers of 3x3 filters throughout the entire network with the stride of 1 pixel and maxpool layer of 2x2 of stride 2. The entire architecture comprises follows this order of convolutional and maxpool layers. Finally, the output will be generated by a softmax layers which is followed by 2 fully connected layers. The 16 in VGG 16 represents the 16 layers that have weights.

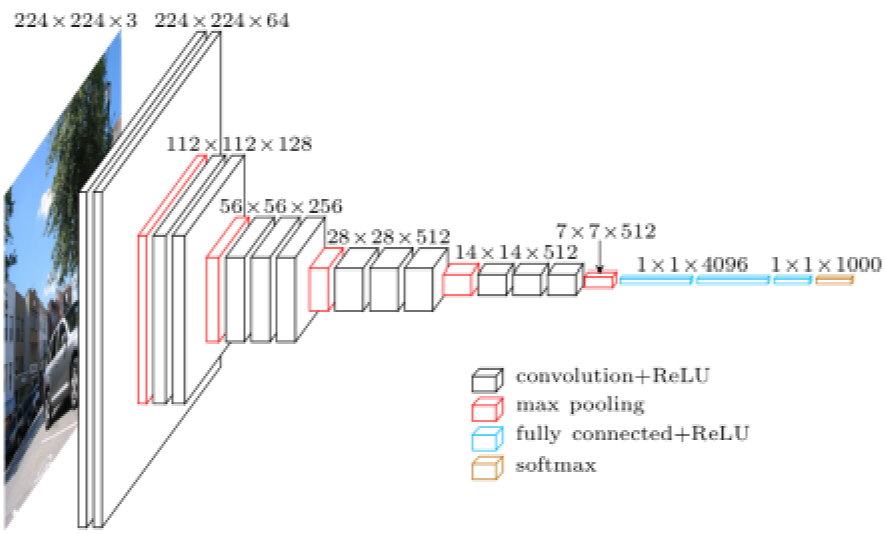

Figure 3 - VGG 16 Architecture

The dataset has been downloaded from Kaggle and saved in the relevant drive folder

```python
from google.colab import drive
drive.mount('/content/gdrive')
```

```python
import os
os.chdir('/content/gdrive/MyDrive')
!wget -c 'https://storage.googleapis.com/kaggle-data-sets/1646010/2702383/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.g
```

```python
import zipfile

zip_ref = zipfile.ZipFile("/content/gdrive/MyDrive/data.zip", "r")
zip_ref.extractall()
zip_ref.close()
```

Figure 4 - Jupyter Notebook Screenprint

Afterwards the required libraries have been imported to the project

```python
import cv2     # for capturing videos
import math   # for mathematical operations
import matplotlib.pyplot as plt    # for plotting the images
%matplotlib inline
import pandas as pd
from keras.preprocessing import image   # for preprocessing the images
import numpy as np    # for mathematical operations
from keras.utils import np_utils
from skimage.transform import resize   # for resizing images
from sklearn.model_selection import train_test_split
from glob import glob
from tqdm import tqdm
```

Python

Figure 5 - Jupyter Notebook Screenprint

Importing the required libraries

Import Libraries

```python
import cv2     # for capturing videos
import math   # for mathematical operations
import matplotlib.pyplot as plt    # for plotting the images
%matplotlib inline
import pandas as pd
from keras.preprocessing import image   # for preprocessing the images
import numpy as np    # for mathematical operations
from keras.utils import np_utils
from skimage.transform import resize   # for resizing images
from sklearn.model_selection import train_test_split
from glob import glob
from tqdm import tqdm
```

Python

Figure 6 - Jupyter Notebook Screenprint

Renaming the images extracted from the dataset and generating a single text file

Rename the images of the dataset according to a project specific naming convention

```python
i = 0
path="/content/gdrive/MyDrive/ASL_Alphabet_Dataset/asl_alphabet_train/Z/"
for filename in os.listdir(path):
  my_dest ="Z_" + str(i) + ".jpg"
  my_source =path + filename
  my_dest =path + my_dest
  os.rename(my_source, my_dest)
  i += 1
```

Python

Creating text files with the images names belonging to each class and collating it into a single text file

```python
f= open("Z.txt","w+")
images = glob("/content/gdrive/MyDrive/ASL_Alphabet_Dataset/asl_alphabet_train/Z/*.jpg")
train_image = []
train_class = []
for i in tqdm(range(len(images))):
    # creating the image name
    train_image.append(images[i].split('/')[7])
    f.write('Z/'+train_image[i]+"\n")
```

Python

```
100%|██████████| 7410/7410 [00:00<00:00, 270052.42it/s]
```

Figure 7 - Jupyter Notebook Screenprint

Assigning images into training and testing datasets

Reading the image name from imageListNew.txt file and assign them to train array list.

```python
# open the .txt file which have names of training images
f = open("/content/gdrive/MyDrive/imageListNew.txt", "r")
temp = f.read()
images = temp.split('\n')

# creating a dataframe having images names
train = pd.DataFrame()
train['image_name'] = images
train = train[:-1]
train.head()
```

Python

Reading the image name from imageListNew.txt file and assign them to test array list.

```python
# open the .txt file which have names of test images
f = open("/content/gdrive/MyDrive/imageListNew.txt", "r")
temp = f.read()
images = temp.split('\n')

# creating a dataframe having images names
test = pd.DataFrame()
test['image_name'] = images
test = test[:-1]
test.head()
```

Python

Figure 8 - Jupyter Notebook Screenprint

Creating tags to identify training and testing images

Creating tag for training and testing images

```python
# creating tags for training image
train_image_tag = []
for i in range(train.shape[0]):
    train_image_tag.append(train['image_name'][i].split('/')[0])

train['tag'] = train_image_tag

# creating tags for test image
test_image_tag = []
for i in range(test.shape[0]):
    test_image_tag.append(test['image_name'][i].split('/')[0])

test['tag'] = test_image_tag
```

Python

Figure 9 - Jupyter Notebook Screenprint

Adding all the training images into a single CSV file

Reading all the images and adding them to a single CSV file.

```python
# getting the names of all the images
train_image = []
train_class = []
for i in tqdm(range(len(images))):
    images_new = glob("/content/gdrive/MyDrive/ASL_Alphabet_Dataset/asl_alphabet_train/"+images[i])

    train_image.append(images_new[0].split('/')[7])

    # creating the class of image
    classes = (images_new[0].split('_')[4].split('/'))
    train_class.append(classes[1])

    # storing the images and their class in a dataframe
    train_data = pd.DataFrame()
    train_data['image'] = train_image
    train_data['class'] = train_class

    # converting the dataframe into csv file
    train_data.to_csv('/content/gdrive/MyDrive/train_new2.csv',header=True, index=False)
```

Python

```
100%|██████████| 2626/2626 [00:00<00:00, 3234.05it/s]
```

Figure 10 - Jupyter Notebook Screenprint

Assigning training images into training array

Read CSV file and assign it to train array.

```python
train = pd.read_csv('/content/gdrive/MyDrive/train_new2.csv')
train.head()
```

Python

Figure 11 - Jupyter Notebook Screenprint

Preprocessing images and converting into an array list,

Converting images into HSV color space, blurring the image using Gaussian blur technique, detecting edges of the image using Sobel edge detection

Preprocessing the images and Converting images to array list

```python
train_image = []

# for loop to read and store frames
for i in tqdm(range(len(images))):
    # loading the image and keeping the target size as (224,224,3)
    img = image.load_img("/content/gdrive/MyDrive/ASL_Alphabet_Dataset/asl_alphabet_train/"+images[i], target_size=(224,224,3))

     # Convert to HSV
     # Converting image color into HSV color space
    img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # Blur the image for better edge detection
    # Bluring the image using Gaussian blur technique
    img_blur = cv2.GaussianBlur(img_hsv, (3,3), 0)

    # Sobel Edge Detection
    # Detecting edges of the image using Sobel edge detection
    sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on the X axis
    sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge Detection on the Y axis
    sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)

    # converting it to array
    img = image.img_to_array(sobely)
    # normalizing the pixel value
    img = img/255
    # appending the image to the train_image list
    train_image.append(img)

# converting the list to numpy array
X = np.array(train_image)

# shape of the array
X.shape
```

Figure 12 - Jupyter Notebook Screenprint

Assigning the VGG 16 model as the base model

Assigning VGG16 model as a base_model

+ Code + Markdown

```python
base_model = VGG16(weights='imagenet', include_top=False)
```

Python

Figure 13 - Jupyter Notebook Screenprint

Reshaping the training and validation frames into a single dimension and normalizing the pixel values

```python
# reshaping the training as well as validation frames in single dimension
X_train = X_train.reshape(2100, 7*7*512)
X_test = X_test.reshape(526, 7*7*512)
```

Python

Normalizing the pixel values

```python
# normalizing the pixel values
max = X_train.max()
X_train = X_train/max
X_test = X_test/max
```

Python

Figure 14 - Jupyter Notebook Screenprint

Defining the model architecture

```python
#defining the model architecture
model = Sequential()
model.add(Dense(1024, activation='relu', input_shape=(25088,)))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(26, activation='softmax'))
```

Python

Figure 15 - Jupyter Notebook Screenprint

Defining a function to save the model with the best weights based on the validation loss value. Then, the model is compiled and the training process will be started.

```python
# defining a function to save the weights of best model
from keras.callbacks import ModelCheckpoint
mcp_save = ModelCheckpoint('/content/gdrive/MyDrive/weight.hdf5', save_best_only=True, monitor='val_loss', mode='min')
```

Python

Compiling the model

```python
# compiling the model
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
```

Python

Training the model

```python
# training the model
model.fit(X_train, y_train, epochs=500, validation_data=(X_test, y_test), callbacks=[mcp_save], batch_size=128)
```

Python

Figure 16 - Jupyter Notebook Screenprint

Results and Discussion

The accuracy and the validation loss of the model is shown in the below diagrams.

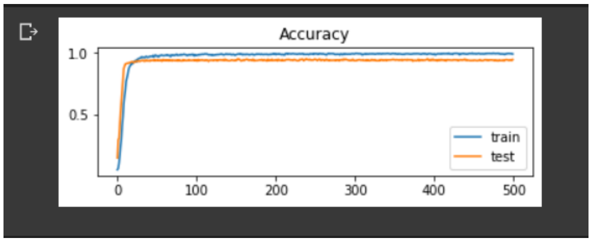

Figure 18 - Classification accuracy

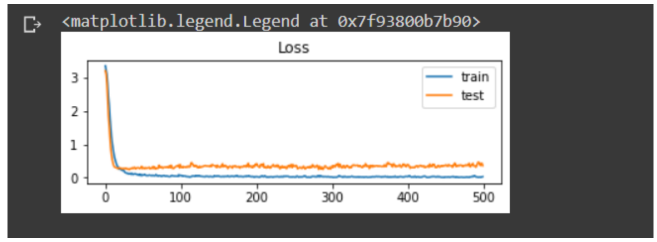

Figure 17 - Validation Loss

The model has been evaluated under the below 4 metrics,

Classification accuracy, precision, recall and F1 score. The below table depicts the values attained for each of these metrics when tested on letter A, S, P, D and W. Additionally, the pre-processing steps have contributed towards improving the accuracy of the predictions.

Letter	**Precision**	**Recall**	**F1 score**
A	1.0	0.95	0.98
S	0.94	0.92	0.93
P	0.89	1.00	0.94
D	1.00	0.74	0.83
W	0.77	0.74	0.75

Figure 19 - Metrics of Model

Limitations and Future Work

As the initial step, the proposed model is developed to identify only the letters of the English alphabet. However, as future work, we can focus on expanding the dataset in order to identify other sign gestures including gestures that involve a movement. In addition, real-time identification of gestures along with facial expression detection can be explored.

Individual Contributions

IT19067902 YOGANATHAN J.A.

1. Understanding the problem and the gathering background information relating to the research problem [4][1][2]
2. Understanding the VGG 16 algorithms and feature extraction and preprocessing techniques that can be incorporated into the algorithm [5]
3. Creating the report and updating the sections of the report
4. Developed a basic UI along with an API to display the prediction of the model
5. Rearranging dataset to suit the VGG 16 algorithm
6. Developing model and training model

IT19004914 JAYASINGHA J.M.M.M

1. Understanding the problem and the gathering background information relating to the research problem [4][1][2]
2. Understanding the VGG 16 algorithms and feature extraction and preprocessing techniques that can be incorporated into the algorithm [5]
3. Creating the report and updating the sections of the report
4. Developed a basic UI along with an API to display the prediction of the model
5. Rearranging dataset to suit the VGG 16 algorithm
6. Developing model and training model

IT19020990 WIJESINGHE M.K

1. Understanding the problem and the gathering background information relating to the research problem [4][1][2]
2. Understanding the VGG 16 algorithms and feature extraction and preprocessing techniques that can be incorporated into the algorithm [5]
3. Creating the report and updating the sections of the report
4. Developed a basic UI along with an API to display the prediction of the model
5. Rearranging dataset to suit the VGG 16 algorithm
6. Developing model and training model

IT19089300 RAMYATHILAKE S.H.M.

1. Understanding the problem and the gathering background information relating to the research problem [4][1][2]
2. Understanding the VGG 16 algorithms and feature extraction and preprocessing techniques that can be incorporated into the algorithm [5]
3. Creating the report and updating the sections of the report
4. Developed a basic UI along with an API to display the prediction of the model
5. Rearranging dataset to suit the VGG 16 algorithm
6. Developing model and training model

OneDrive Link to Demonstration Video

shorturl.at/sDF01

References

K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 4896-4899, doi: 10.1109/BigData.2018.8622141 [1]

Mazhar, Osama, Sofiane Ramdani, and Andrea Cherubini. 2021. "A Deep Learning Framework for Recognizing Both Static and Dynamic Gestures" *Sensors* 21, no. 6: 2227. [2]

https://www.analyticsvidhya.com/blog/2019/09/step-by-step-deep-learning-tutorial-video-classification-python/ [3]

https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss [4]

https://www.youtube.com/watch?v=mjk4vDYOwq0&t=286s [5]