Project report on

Sign Language Translator

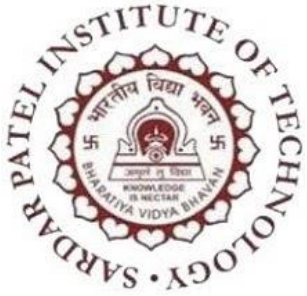
BACHELOR
IN

Electronics Engineering

Submitted By

Lance Fernandes
2017110015
Manav Jain
2017110021
Siddhant Adhikari
2017110001

Bharatiya Vidya Bhavans
Sardar Patel Institute of Technology



Department of Electronics Engineering Sardar Patel Institute of Technology Munshi Nagar, Andheri(W), Mumbai-400058
UNIVERSITY OF MUMBAI 2019-2020

I. Introduction

Sign language is a language through which communication is possible without the means of acoustic sounds. Instead, sign language relies on sign patterns, i.e., body language, orientation and movements of the arm to facilitate understanding between people. It has been estimated that there are between 0.9 and 14 million hearing impaired in India and perhaps "one of every five people who are deaf in the world, lives in India", making it the country with the largest number of Deaf, and perhaps also the largest number of sign language users. Keeping this in mind we aim to build a convolutional neural network model that classifies all the alphabetical gestures correctly for better interpretation and effective communication.

II. Abstract

It has been estimated that there are more than a million hearing impaired in India making the country with the largest number of deaf and mute and perhaps the country with the largest number of sign language users. There exists a communication gap between the mute-hearing people with normal people which is of paramount importance to be bridged.The aim is to create a highly economical, accurate and independent model which will translate sign language gestures into text as per American Sign Language standard.The model in this project is built using the tools provided by tensorFlow and Keras Python libraries.

III. Literature Survey

There are two approaches used in sign language recognition they are sensor based and image based. In sensor based approach, sensor collects data by performing various gestures then the data is analysed based on recognition model. In vision based approach the data is collected in the form of images then extraction of main features of image is done to recognise the gesture .In recent research sign language recognition system has three important steps data acquisition , extraction of data and classification .For recognition , classification methods used are Neural network ,Support vector machine ,Hidden Markov model ,scale invariant feature transform.

In the research paper mentioned in Ref 1 a sensor base approach is used. A flex sensor based model is developed and a text to speech synthesizer based on HMM is used for text to speech conversion. In research paper in ref 2 a flex sensor and accelerometer are used for gesture classification. The set of gestures are very small. The gesture is further displayed on LCD screen. In the paper mentioned in ref 4 a neural network model is used for image classification and the accuracy is no more than 85%. In Ref 7 flex sensors and accelerometer is used and a simple machine learning model is applied for classification.

IV. Project Objective

- Create a Model that mitigates the challenges faced by deaf and mute people.
- Pre-process the data to improve the accuracy of the model.
- Classify and correctly predict the 26 alphabets in sign language.

V. Methodologies

We have used convolutional neural network to train our model. We have used an existing dataset to train out model. We have used The RestNet 50 model to improve the accuracy of our model. The data is Pre-Processed and augmented for improving the versatility of the model. The data is then tested and evaluated on a different set of images.

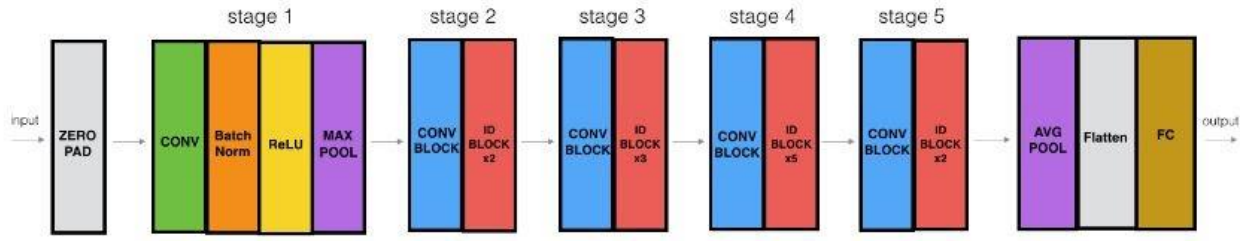
Code Snippets of our model:

```
def convolutional_block(X, f, filters, stage, block, s = 2):
    """
    Implementation of the convolutional block as defined in Figure 4

    Arguments:
    X -- input tensor of shape (m, n_H_prev, n_W_prev, n_C_prev)
    f -- integer, specifying the shape of the middle CONV's window for the main path
    filters -- python list of integers, defining the number of filters in the CONV layers of the main path
    stage -- integer, used to name the layers, depending on their position in the network
    block -- string/character, used to name the layers, depending on their position in the network
    s -- Integer, specifying the stride to be used

    Returns:
    X -- output of the convolutional block, tensor of shape (n_H, n_W, n_C)
    """

    # defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    # Retrieve Filters
    F1, F2, F3 = filters

    # Save the input value
    X_shortcut = X


    ##### MAIN PATH #####
    # First component of main path
    X = Conv2D(F1, (1, 1), strides = (s,s), name = conv_name_base + '2a', padding = 'valid', kernel_initializer = glorot_uniform
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    ### START CODE HERE ###
```

```
##### MAIN PATH #####
# First component of main path
X = Conv2D(F1, (1, 1), strides = (s,s), name = conv_name_base + '2a', padding = 'valid', kernel_initializer = glorot_uniform(
X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
X = Activation('relu')(X)

### START CODE HERE ###

# Second component of main path (≈3 lines)
X = Conv2D(F2, (f,f), strides = (1,1), name = conv_name_base + '2b', padding = 'same', kernel_initializer = glorot_uniform(se
X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
X = Activation('relu')(X)

# Third component of main path (≈2 lines)
X = Conv2D(F3, (1,1), strides = (1,1), name = conv_name_base + '2c', padding = 'valid', kernel_initializer = glorot_uniform(s
X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

##### SHORTCUT PATH #### (≈2 lines)
X_shortcut = Conv2D(F3,(1,1),strides = (s,s), name = conv_name_base + '1',padding = 'valid', kernel_initializer = glorot_unif
X_shortcut = BatchNormalization(axis=3,name = bn_name_base + '1')(X_shortcut)

# Final step: Add shortcut value to main path, and pass it through a RELU activation (≈2 lines)
X = Add()([X,X_shortcut])
X = Activation('relu')(X)

### END CODE HERE ###

return X
```

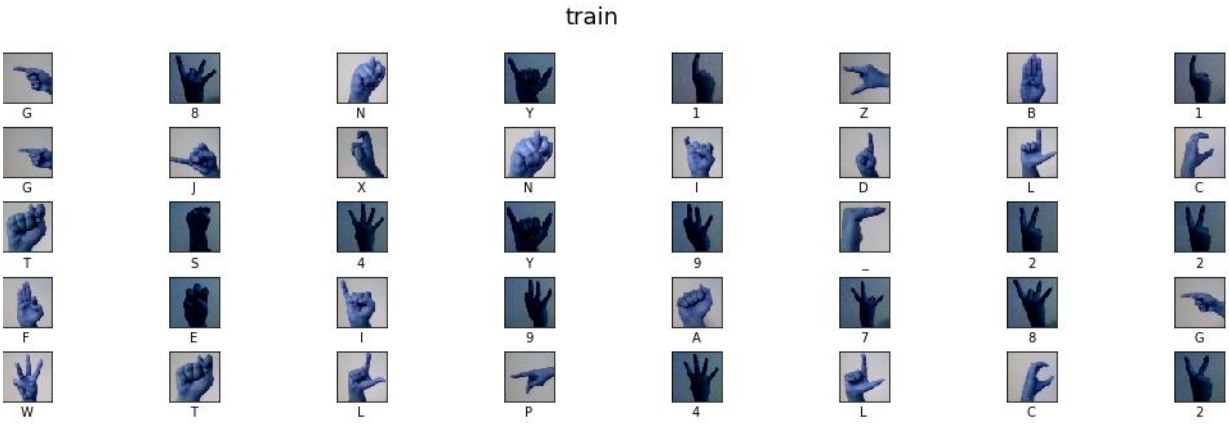
```
X_input = Input(input_shape)


# Zero-Padding
X = ZeroPadding2D((10, 10))(X_input)

# Stage 1
X = Conv2D(64, (7, 7), strides = (2, 2), name = 'conv1', kernel_initializer = glorot_uniform(seed=0))(X)
X = BatchNormalization(axis = 3, name = 'bn_conv1')(X)
X = Activation('relu')(X)
X = MaxPooling2D((3, 3), strides=(2, 2))(X)

# Stage 2
X = convolutional_block(X, f = 3, filters = [64, 64, 256], stage = 2, block='a', s = 1)
X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')
X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')

### START CODE HERE ###

# Stage 3 (≈4 lines)
X = convolutional_block(X, f = 3, filters = [128,128,512], stage = 3, block = 'a', s = 2)
X = identity_block(X, 3, [128,128,512], stage = 3, block = 'b')
X = identity_block(X, 3, [128,128,512], stage = 3, block = 'c')
X = identity_block(X, 3, [128,128,512], stage = 3, block = 'd')

# Stage 4 (≈6 lines)
X = convolutional_block(X, f = 3, filters = [256,256,1024], stage = 4, block = 'a', s = 2)
X = identity_block(X, 3, [256,256,1024], stage = 4, block = 'b')
X = identity_block(X, 3, [256,256,1024], stage = 4, block = 'c')
X = identity_block(X, 3, [256,256,1024], stage = 4, block = 'd')
X = identity_block(X, 3, [256,256,1024], stage = 4, block = 'e')
X = identity_block(X, 3, [256,256,1024], stage = 4, block = 'f')

# Stage 5 (≈3 lines)
X = convolutional_block(X, f = 3, filters = [512,512,2048], stage = 5, block = 'a', s = 2)
X = identity_block(X, 3, [512,512,2048], stage = 5, block = 'b')
X = identity_block(X, 3, [512,512,2048], stage = 5, block = 'c')

# AVGPOOL (≈1 line). Use "X = AveragePooling2D(...)(X)"
X = AveragePooling2D(pool_size = 2 ,name= 'avg_pool')(X)
```

Structure of the model



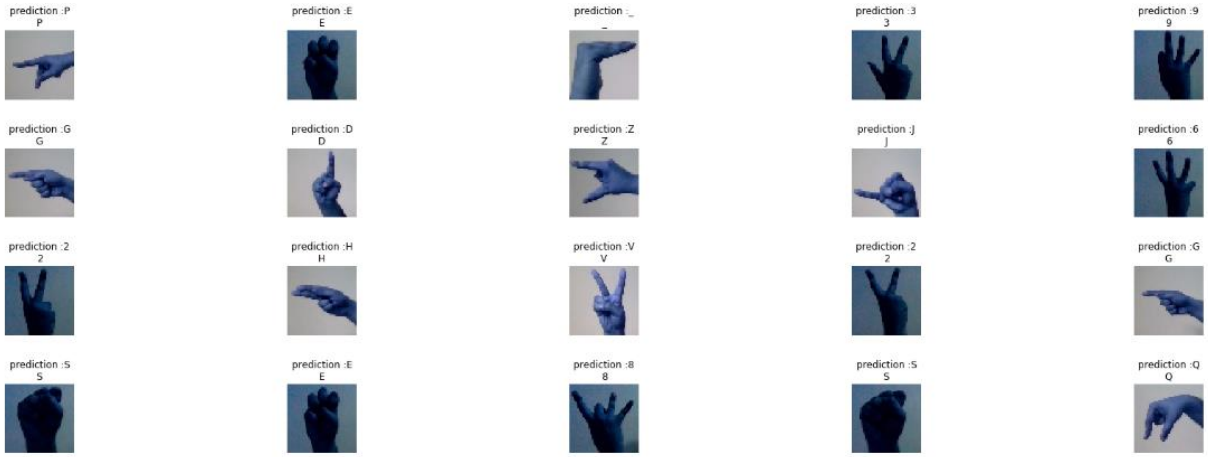
The reason to use RestNet 50 model is during each iteration of training a neural network, all weights receive an update proportional to the partial derivative of the error function with respect to the current weight. If the gradient is very small then the weights will not be change effectively and it may completely stop the neural network from further training. The phenomenon is called vanishing gradients. More specifically we can say that the data is disappearing through the layers of the deep neural network due to very slow gradient descent.

Screenshot of our dataset mentioned in References

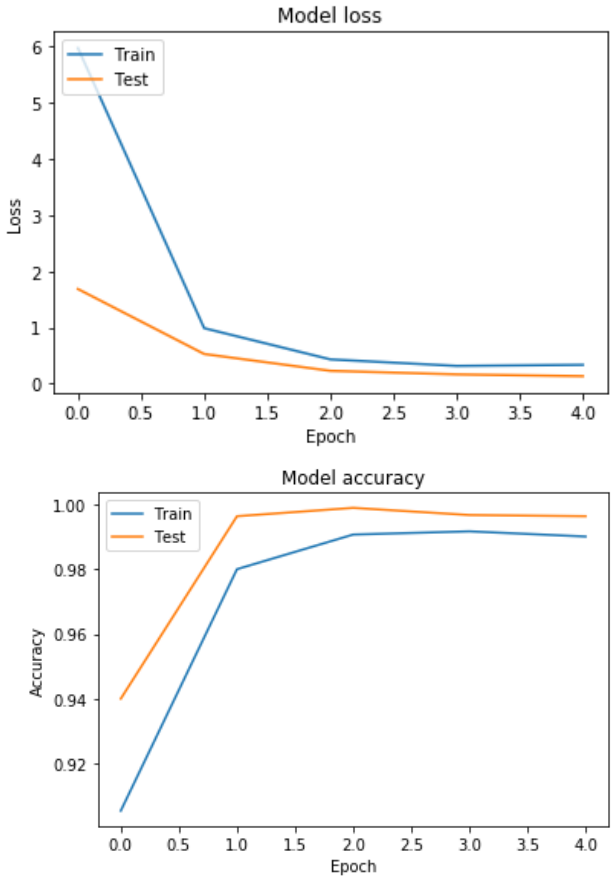


VI. Results

Output Predictions:



Accuracy Results:



VII. Conclusion:

- The idea of communication of speech impaired through sign language is discussed and the problem they face when there is absence of interpreter.
- This project aims to implement a sign language translator which will facilitate them to communicate with non-signers without any complications. They can also communicate in the absence of a sign language interpreter.
- The focus of this project is to automate sign language interpretation using deep learning nets for improving the prediction accuracy to a satisfactory level.

VIII. References:

[1] P. Vijayalakshmi and M. Aarthi, "Sign language to speech conversion," 2016 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, 2016, pp. 1-6, doi: 10.1109/ICRTIT.2016.7569545.Shaheer Bin Rizwan, Muhammad Saad Zahid Khan, American Sign Language Translation via Smart wearable technolgy,IEEE,2019.

[2] Shahrukh Javed, Ghousia Banu, Suganthi Kani, Wireless Glove for Hand Gesture Acknowledgement, Robotics and Automation Journal,2018.

[3] S. He, "Research of a Sign Language Translation System Based on Deep Learning," 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), Dublin, Ireland, 2019, pp. 392-396, doi: 10.1109/AIAM48774.2019.00083.

[4] T. Karayılan and Ö. Kılıç, "Sign language recognition," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, 2017, pp. 1122-1126, doi: 10.1109/UBMK.2017.8093509.

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

[6] S. B. Rizwan, M. S. Z. Khan and M. Imran, "American Sign Language Translation via Smart Wearable Glove Technology," 2019 International Symposium on Recent Advances in Electrical Engineering (RAEE), Islamabad, Pakistan, 2019, pp. 1-6, doi: 10.1109/RAEE.2019.8886931.

[7] Tecperson, Sign Language MNIST, 2017, Version 1, https://www.kaggle.com/datamunge/sign-language-mnist

[8] Akash, ASL Alphabet, 2018, Version 1, https://www.kaggle.com/grassknoted/asl-alphabet