4/21/2020

Bangla Sign language Detection Web Application using CNN Model & Posenet

Project Report V1.0

Project Member: Tabashum, Thasina | Badruddoja, Syed Project Adviser: Dr. Ting Xiao

CSCE5300 | Introduction to Big data | SPRING 2020 | University Of North TExas

Table of Contents

[Abstract 3](#_Toc38364672)

[Introduction 3](#_Toc38364673)

[Project Organization & Team 4](#_Toc38364674)

[Related projects 4](#_Toc38364675)

[Project Design and Milestones: 5](#_Toc38364676)

[Preprocessing 5](#_Toc38364677)

[Training Model 6](#_Toc38364678)

[Figure 3:Convolutional neural Network With VGG16[3] 6](#_Toc38364679)

[Figure 4:VGG 16 with 22 layers and 16 weights [3] 6](#_Toc38364680)

[Application Development 7](#_Toc38364681)

[Tools & Libraries 7](#_Toc38364682)

[Table: Software Packages in Use 8](#_Toc38364683)

[Dataset 8](#_Toc38364684)

[Table 2: Class and training set count 8](#_Toc38364685)

[Project Plan: 8](#_Toc38364686)

[Table 3: Project Plan 9](#_Toc38364687)

[Implementation 9](#_Toc38364688)

[Installation Guide: 9](#_Toc38364689)

[Figure 6: installation of the web application 10](#_Toc38364690)

[Software Building process: 10](#_Toc38364691)

[Figure8 : Prediction function 11](#_Toc38364692)

[Demonstration 12](#_Toc38364693)

[Figure 4: Web View. Before uploading the image 12](#_Toc38364694)

[Figure 5: After Uploading the a picture 12](#_Toc38364695)

[Results: 12](#_Toc38364696)

[Graph1: Loss comparison of training set and validation set 13](#_Toc38364697)

[Graph 2: Accuracy Comparison of Training Set & Validation Set 13](#_Toc38364698)

[VGG16: 13](#_Toc38364699)

[Graph 3: VGG16 model loss 14](#_Toc38364700)

[Graph 4: VGG16 model accuracy 14](#_Toc38364701)

[Proposed model : 15](#_Toc38364702)

[Table 4:  Model Summary 15](#_Toc38364703)

[Table 5 : Layer which are removed 16](#_Toc38364704)

[Graph 5 : Accuracy of proposed model 16](#_Toc38364705)

[Graph 6: Loss of proposed model 17](#_Toc38364706)

[Difference with the existing project using the same dataset: 17](#_Toc38364707)

[Difference on Model Stage: 17](#_Toc38364708)

[Table 3: Average Accuracy Comparison for CNN-VGG16 Model 18](#_Toc38364709)

[Conclusion 18](#_Toc38364710)

[References 18](#_Toc38364711)

# Abstract

With the advent of artificial intelligence and machine learning, the classification tasks have become superfast in many cases. In this project we have explored classification of sign language images with various neural network technologies such as VGG16( Visual geometry Group) and modified VGG16 which are variants of convolutional neural network. We also developed a web application with nodejs for users to upload images and classify sign language.

# Introduction

The image classification task is one of the most wanted task in artificial intelligence and machine learning world. Throughout the deep learning course, we have learnt various methods of classification staring from CNN, RNN, LSTM etc. The most common of all the neural network approach to image classification is CNN because of the convolutions performed on the image to get the latent variables. The CNN has many variations in its own like VGG, Alexnet, Inception V1, Inception V2 etc. Each one has it’s own working scheme and learning advantages in terms of input, output and optimization parameter.

# Project Organization & Team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Member | Email | Responsibility | Meeting Schedule | Team Adviser |
| 1 | Tabashum, Thasina | [thasinatabashum@my.unt.edu](mailto:thasinatabashum@my.unt.edu) | Coding/Analysis/Application Development | Twice in a week | Dr. Ting Xiao |
| 2 | Badruddoja, Syed | [syedbadruddoja@my.unt.edu](mailto:syedbadruddoja@my.unt.edu) | Analysis/Documentation/Presentation development |

Project Repository: <https://drive.google.com/open?id=1c_hwa7WDfklc69cntgO1mAA1xvS-cCLLp05LoVJyMu8>

# Related projects

There are several projects/research that work on pose estimation and sign language prediction. We have picked three such studies to discuss out of all.

1. Real-time Human Pose Estimation in the Browser with TensorFlow.js [Link: <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5>]
2. Sign Language Recognition using 3D convolutional neural networks:[Link: <https://ieeexplore.ieee.org/abstract/document/7177428>]
3. A Real-Time System for Recognition of American Sign Language by using Deep Learning[Link: <https://ieeexplore.ieee.org/document/8441304>]

.

In one of the paper [4], they have proposed a novel 3D convolutional neural network (CNN) which extracts discriminative spatial-temporal features from raw video stream automatically without any prior knowledge, avoiding designing features. ​Also, to boost the performance, multi-channels of video streams, including color information, depth clue, and body joint positions, are used as input to the 3D CNN in order to integrate color, depth and trajectory information​.

Another paper discusses a Real-Time System for Recognition of American Sign Language by using Deep Learning[[5]](https://ieeexplore.ieee.org/document/8441304)​. In this study, the convolutional neural network was trained by using dataset collected in 2011 by Massey University, Institute of Information and Mathematical Sciences, and 100% test accuracy was obtained. ​

# Project Design and Milestones:

Sign language detection is a common neural network problem that requires image analysis with different images and classes. What makes our project different then all the projects in the publication forum is that we would put up a study of computation time and accuracy comparison for detecting Bangla sign language with respect to features that may vary in size. Also, the application.

## Design Methodology

Following are the main design methods which we will follow in the process of this project.

* Detecting Hand from the image using PoseNet and create a reasonable bounding box
* Feeding ConvNet the resized image dataset
* Feeding the ConvNet the full image
* Compare the two results in terms of accuracy and computational time
* Finally a simple web version to test in real time.



Figure 1: Bangla sign language (Request, Today and Cough)

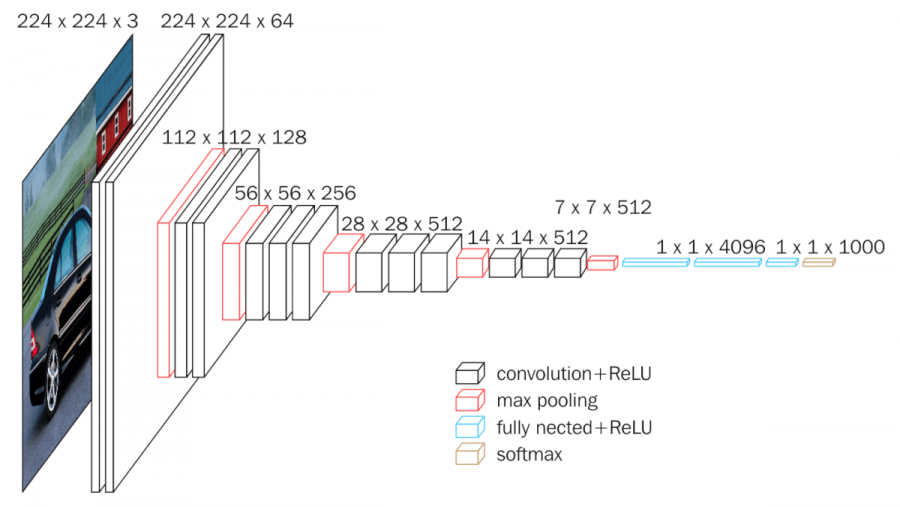
### Preprocessing

[to be edited by Thasina]

### Training Model

**Transfer Learning[]**

**Convolutional neural network** is one of the known deep learning technique to perform image classification, recognition, object recognition, face recognitions and many more. Convolutional neural net takes an input, process it and then performs training of pixels into a model and predicts the images as an object or a class. The features in the image can vary for different sizes like 6,6,3 which implies that height =6, width = 6 and there are 3 colors or channels RGB. This is the simplest image we can ever have but in the modern world the images are of high resolution which means the pixel count can be of millions. Convolutional neural network works in two phases. First phases is learning and second phase is classifying. In the learning phase there are different layers of operations like convolutional network, RELU activation and maxpooling. In the classification phase, the data is flattened and then fed into the normal neural network architecture.



### Figure 3:Convolutional neural Network With VGG16[3]

**VGG16** is one of the convolutional neural architecture that implies 16 layers that consists of Convolutional layers, Max Pooling layers, Activation layers, Fully connected layers.



### Figure 4:VGG 16 with 22 layers and 16 weights [3]

The VGG-16 architecture contains 13 convolutional layers, 5 max pooling layers and 3 dense layers which sums up to 21 layers but only 16 weights. Layer 1 has 64 filters, layer2 has 128, layer 3 with 256, layer 4 and 5 with 512 filters. VGG16 was trained with imagenet dataset which has over 14 million images and 1000 classes and also has the ability to achieve more than 90% of accuracy[3] .

### Application Development

A browser-based application is planned to be created to for real time recognition of sign languages that will have the deep learning framework for training and testing purposes. The training images will be used to train the model and there would be separate testing images for prediction purposes.

## Tools & Libraries

We require a list of tools of libraries that are part of this package. We will be reading the data, perform preprocessing, training the neural network, and then testing in real time based on the image captured. Currently following are the list of packages planned for usage.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Package/ Framework/ Software | Definition | Purpose |
| 1 | Tensorflow | End to end system of libraries for machine learning | Provide Library Framework |
| 2 | Keras | Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](https://github.com/tensorflow/tensorflow) | Provide Neural network API |
| 3 | Numpy | NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices | Data Processing |
| 4 | Pandas | a software library written for the Python programming language for data manipulation and analysis. | Data processing |
| 5 | matplotlib | Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy | Data Visualization |
| 6 | Nodejs | Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a web browser | Browser based application |

### Table: Software Packages in Use

## Dataset

We 100 pictures for each classifications and total 8 types of sign language. So, in total we had 800 images for this for training and testing purpose. 651 images for training, 78 images for validation and 81 images for testing.

### Dataset Preprocessing:

We proposed to use posenet for preprocessing. But after doing a lot of hand engineering it seems difficult to obtain an area with that rather using CNN model is a lot more easier. So we used VGG16 preprocessing.

Some explanation of VGG16 preprocessing from paper or online blog]

## Project Plan:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Task | Description | Work Type | Status |
| 1 | Perform Literature Survey | Find articles of work on similar projects on IEEE | Documentation/ Survey | Ongoing |
| 2 | Preprocess Data | Process image data to feed into neural network | Coding/Planning | Done |
| 3 | Create program | Starting implementation with initial package loading | Coding | Done |
| 4 | Load Modified VGG16 Model | Difference Convolution, Max pooling and relu activation combination | Coding/ Accuracy Study | Done |
| 6 | Train model with Training Data | Train model with training data set and model compile option | Coding/ Accuracy Study | Done |
| 7 | Test validation Set of Data | Test the Trained Model with validation data | Coding/ Accuracy Study | Done |
| 8 | Predict Test set data | Predicting test data | Coding/Accuracy Study | Done |
| 9 | Prepare Project Update | Project Update Work on status of the project | Documentation | Done |
| 10 | Prepare Final Report | Final Project report with all the results, findings and observation | Documentation | Done |

### Table 3: Project Plan

# Implementation

The implementation of this project had different segments which are neural network training and testing part and web application part. Following is user guide for installation guide if anybody requires to implement.

## Installation Guide:

To run the application unzip the file.

Step 1: Install node.js according to your OS. <https://nodejs.org/en/download/>

Step 2: Unzip the folder. There will be a local server folder.

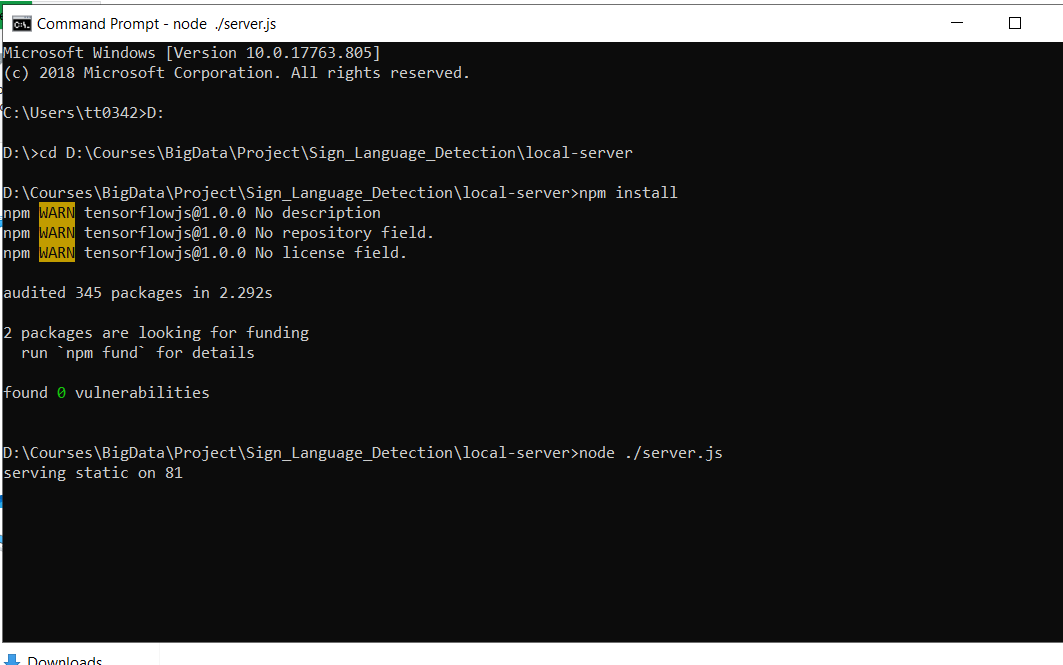
Step 3: Go to command prompt

Step 4: Run npm install [ Make sure to go local server folder and run  because package.json is written there]

Step 5: node ./server.js [  this will start local server at 81 port, you can change the port on server.js file]

Note: Now our node application is ready and local server

Step 6: If running on 81 port then url to run the application is <http://localhost:81/predict-with-tfjs.html>

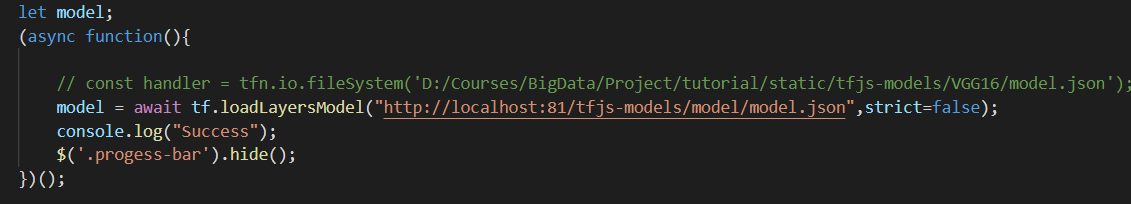


### Figure 6: installation of the web application

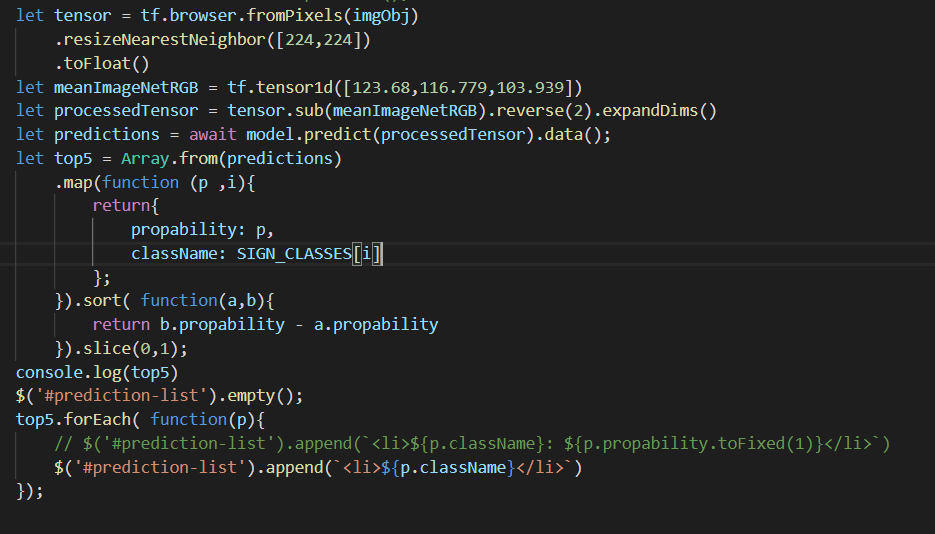
## Software Building process:

Development Process in details:

* **Building model and saving it:** we built the model and trained it using tensorflow and keras. And saved it as an h5 file.
* **Converting tensorflow model:** First, we install using pip install tensorflowjs from the terminal. Then we ran on command prompt:
  + tensorflowjs\_converter --input\_format location/model.h5 locationtosavetheconversion
  + There will be multiple files after conversion so it is suggested to save the conversion on a folder
* Installed nodejs
* On our main folder there two subfolders -> 1)local server 2)static
* Local server folder includes the package.json file and server.js file
* Static folder has tfjs-models subfolder navigating inside where we saved our models as sub directories.Each of the directories contains model.json and corresponding weight files for each model
* On the static folder we have three files which are predict-with-tfjs.html, sign\_classes.js and Predict.js
* **predict-with-tfjs.html** : This html files is the front end design of the web application
* **Sign\_classes.js:** includes the predictions labels
* **Predict.js:** here we load the model and preprocess the input image and do the prediction



#### Figure 7: loading the model



### Figure8 : Prediction function

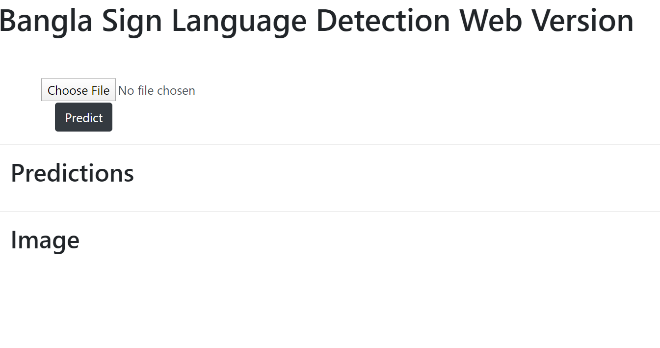
When the user click the predict button we get the image from the #selected-image element, then we transformed the image as the model expected. The model expect rank 4 tensor object with the height and width dimensions of 224\* 224. For further preprocessing we followed the pre-process of VGG16 model as explained on their paper. Currently tensorflow-js does not have this preprocessing function. On this stage we also tried to use posenet for preprocessing the input but we did not achieve what we intended to do.

VGG16 preprocessing: They subtract the mean RGB value, computed on Imagenet training set from each pixel.

We did the same preprocessing on this section of the code.

## Demonstration

Following is the demonstration of sign language image recognition from the website. A user has to upload image and then the web application will predict the image with the particular sign language.



### Figure 4: Web View. Before uploading the image

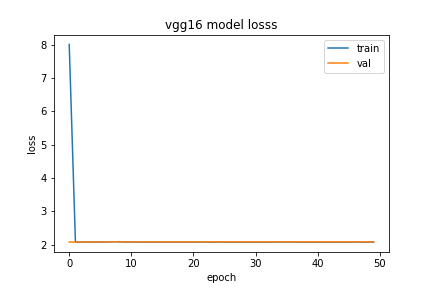


### Figure 5: After Uploading the a picture

# Results:

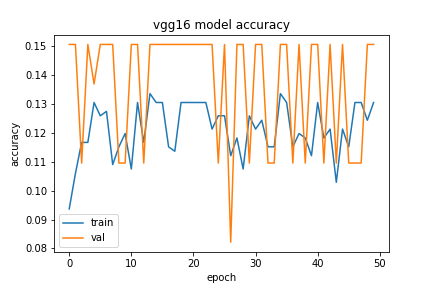
Following are the metrics of loss and accuracy with respect to model used in the experiment, Firstly the basic VGG16 model is used to train the model with the training dataset. The loss from Graph 1 shows the level of loss to be above 2 constantly for 50 epochs of training of data.

## VGG16:



### Graph 1: VGG16 model loss

Graph 2 shows the accuracy of running VGG16 on the training data. The accuracy is very low ranging from 9perent to 15 percent. The training data accuracy is between 9% to 13% and the validation data accuracy is ranges between 8% to 15%.



### Graph 2: VGG16 model accuracy

## Proposed model :

Model: "sequential\_10"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type)                 Output Shape Param #

=================================================================

conv2d\_70 (Conv2D)           (None, 224, 224, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_71 (Conv2D)           (None, 224, 224, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_26 (MaxPooling (None, 112, 112, 64)      0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_72 (Conv2D)           (None, 112, 112, 128) 73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_73 (Conv2D)           (None, 112, 112, 128) 147584

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_27 (MaxPooling (None, 56, 56, 128)       0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_74 (Conv2D)           (None, 56, 56, 256) 295168

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_75 (Conv2D)           (None, 56, 56, 256) 590080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_76 (Conv2D)           (None, 56, 56, 256) 590080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_28 (MaxPooling (None, 28, 28, 256)       0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_10 (Flatten)         (None, 200704) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_16 (Dense)             (None, 256) 51380480

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_17 (Dense)             (None, 128) 32896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_18 (Dense)             (None, 8) 1032

=================================================================

Total params: 53,149,896

Trainable params: 53,149,896

Non-trainable params: 0

### Table 4:  Model Summary

We reused the VGG16 model but removed some layers of the model as our dataset is small and our classification number is also small compared to imagenet classification.

We removed the layers in table 5

conv2d\_84 (Conv2D)           (None, 28, 28, 512) 1180160

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_85 (Conv2D)           (None, 28, 28, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_86 (Conv2D)           (None, 28, 28, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_32 (MaxPooling (None, 14, 14, 512)       0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_87 (Conv2D)           (None, 14, 14, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_88 (Conv2D)           (None, 14, 14, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_89 (Conv2D)           (None, 14, 14, 512) 2359808

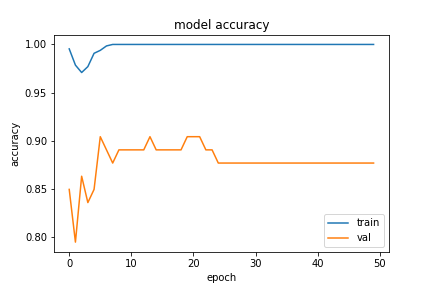
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_33 (MaxPooling (None, 7, 7, 512)         0

0

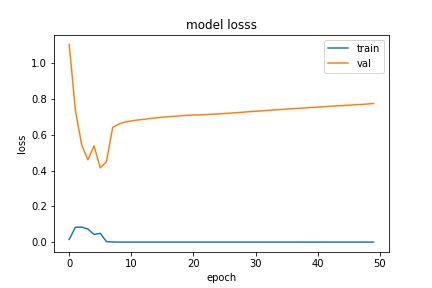
### Table 5 : Layer which are removed

Following is the graph 3 of accuracy with proposed model. The training data has a accuracy of close to 99% and validation data has an average accuracy of 87%.



### Graph 3 : Accuracy of proposed model

Similarly, the loss of training data has gone very low to the point of 0.001 approximation And validation data settled on loss range of 0.6 to 0.8. We can see this from Graph 4.



### Graph 4: Loss of proposed model

## Difference with the existing project using the same dataset:

Difference on Preprocessing Stage: Previous work the model trained with a cropped gray scale image. Whereas we trained different sizes of images with 3 channels with full image.

### Difference on Model Stage:

[Compare the used model and old model]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation of Layer | Earlier Output Shape | Current Output Shape | Earlier  Kernel size | Earlier  Parameters | Current Parameter |
| Convolution(Conv2D) | ( 64, 64, 32) | 224,224,64 | 3 x 3 | 320 | 1792 |
| Convolution(Conv2D) | ( 64, 64, 32) | 224,224,64 | 3 x 3 | 9248 | 36928 |
| Max-pooling | (32, 32, 32) | 112,112,64 | 2 x 2 | 0 | 0 |
| Convolution(Conv2D) | ( 32, 32, 32) | 112,112,128 | 3 x 3 | 18496 | 79856 |
| Convolution(Conv2D) | ( 32, 32, 32) | 112,1112,128 | 3 x 3 | 36928 | 147854 |
| Max-pooling | ( 16, 16, 64) | 56,56,128 | 2 x 2 | 0 | 0 |
| Convolution(Conv2D) | NA | 56,56,256 |  |  | 295168 |
| Convolution(Conv2D) | NA | 56,56,256 |  |  | 590080 |
| Convolution(Conv2D | NA | 56,56,256 |  |  | 590080 |
| Max-pooling | NA | 28,28,256 |  |  | 0 |
| Flatten | (16384) | 200704 | N/A | 0 |  |
| Fully connected | (128) | NA | N/A | 2097280 |  |
| Dropout | ( 128) | NA | N/A | 0 |  |
| Fully connected | ( 10) | NA | N/A | 1290 |  |
|  |  |  |  |  |  |

### Table 6 : Comparison Chart

Total parameters: 21, 63, 562

Total trainable parameters: 21, 63, 562

Non-trainable parameters: 0

# Conclusion

We used a VGG16 model but decreased the network depth. In detail will be discussed. Accuracy of the proposed model on the test set is 95.06% and Validation accuracy 87.67%. A Web Application Developed a node js application. Where the user will upload a picture and It will predict the sign language. We tried to use posenet to preprocess the image. We intend to crop the image on the area it was needed but after trying to implement that stage, we have realized that it will take a lot of hand engineering for feature extraction. It is difficult to come up     with every scenario how a person can be situated and then using the coordinates of the poses to extract the area we needed. Transfer learning performed to vgg16 , resulted in low accuracy, due to low dataset quantity VGG16 is for 1000 image classification which did not result well, changed last layer of vgg16 as only 8 output and vgg16 has 1000 output.

# References

1. Understanding of Convolutional Neural Network (CNN) — Deep Learning - [<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>]
2. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53]
3. CNN Architecture Series — VGG-16 with implementation(Part I)[https://medium.com/datadriveninvestor/cnn-architecture-series-vgg-16-with-implementation-part-i-bca79e7db41
4. Sign Language Recognition using 3D convolutional neural networks - <https://ieeexplore.ieee.org/abstract/document/7177428>​
5. A Real-Time System for Recognition of American Sign Language by using Deep Learning - <https://ieeexplore.ieee.org/document/8441304>