A REPORT

ON

CLASSIFICATION OF AMERICAN SIGN LANGUAGE INPUTS BY USING OPENCV AND CONVOLUTIONAL NEURAL NETWORKS

By

Yash Bhagat	2017A7PS0063P	B.E. CSE
Nayan Khanna	2017B4A70636P	M.Sc. Maths + B.E. CSE

Prepared in partial fulfilment of the

Practice School-I Course No.

BITS F221/BITS F231/BITS F241

AT

IDS Infotech Ltd.

IT Park, Chandigarh, India

A Practice School-I station of

Birla Institute of Technology & Science, Pilani

(July, 2019)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)
Practice School Division

Station: IDS Infotech
Centre: IT Park, Chandigarh
Duration: From: 21st May 2019
To:13th July 2019
Date of Submission: 12th July 2019

Title of the Project: CLASSIFICATION OF AMERICAN SIGN LANGUAGE INPUTS BY USING OPENCV AND NEURAL NETWORKS

2017A7PS0063P	Yash Bhagat	B.E.CSE
2017B4A70636P	Nayan Khanna	M.Sc. Maths + B.E.CSE

Name of Expert: Mr. Anirudh Munj
Designation: ML Engineer, IDS Infotech
Name of the PS Faculty : Dr. Rajeev Taliyan

Key Words: OpenCV, Keras, Python, Image Processing, Python, Neural Networks, Feature Detection, Machine Learning, Computer Vision
Project Area: Image Processing, Neural Networks, Machine Learning

Abstract: The project CLASSIFICATION OF AMERICAN SIGN LANGUAGE INPUTS was inspired by various gesture recognition projects and the motive to devise a method to help other people communicate using American Sign Language. The project makes use of various python libraries and operations to convert ASL images of hands and converts it into an edged image for classification by the software. The classifier used in this project makes use of Computer Vision and Neural Networks in order to understand and learn from a huge dataset of labelled images of ASL. The model achieved an accuracy of 90+% while classifying the test dataset. The project was deployed over localhost using flask and users could upload images and check the corresponding label. Further we are planning to develop this into an app for the mobile for the common people to use and see the results in real time using webcam.

Signature of the Students:
Date: 12th July 2019

Signature of PS Faculty:
Date:

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our mentor Mr. Anirudh Munj (ML Engineer, IDS Infotech) as well as our PS instructor Dr.Rajeev Taliyan (Associate Professor, BITS Pilani) who gave us the opportunity to be a part of IDS Infotech as summer trainees and work on Image Processing and Neural Networks. Mr. Anirudh has helped us a lot regarding this project by providing resources and guidance to us. We have learned many new things while working on this project for which we are really thankful.

Secondly we would also like to thank our parents, friends and Ms. Paridhi Kashyap (Student co-instructor) for their constant support and guidance.

ABSTRACT

The project CLASSIFICATION OF AMERICAN SIGN LANGUAGE INPUTS was inspired by various gesture recognition projects and the motive to devise a method to help other people communicate using American Sign Language. The project makes use of various python libraries and operations to convert ASL images of hands and converts it into an edged image for classification by the software. The classifier used in this project makes use of Computer Vision and Neural Networks in order to understand and learn from a huge dataset of labelled images of ASL. The model achieved an accuracy of 90+% while classifying the test dataset. The project was deployed over localhost using flask and users could upload images and check the corresponding label. Further we are planning to develop this into an app for the mobile for the common people to use and see the results in real time using webcam.

INTRODUCTION

About the Company:

For the first week of our PS-1, we focused on getting to know more about our organisation, IDS Infotech Ltd. and the domains it currently works in. Established in 1989, IDS Infotech Ltd. (IDS) now had 1000+ employees working around the clock. Also it has been a preferred Business Process Management and Software Solution provider to Fortune 500 companies and AM100 law firms worldwide for over two decades. It has various delivery centers in North India, multiple front-end offices globally, including the US, UK and the Netherlands.

The Problem and Its Background:

Sign language is a form of manual communication that involves conveying ideas through hand gestures, movement and orientation of the body, and facial expressions. It is used by people with either hearing and/or speaking impairments which hinder them using verbal communication. A communication barrier is formed between the majority and the people who depend on sign language for communication since the majority do not understand sign Language. So, our motive of this project was to devise a method to help other people communicate using American Sign Language. We made a portal where users could upload images of American Sign Language alphabet gestures and see the corresponding alphabet.

THE PROCESS

1. Preprocessing of Images and the Dataset

All the images were captured in 3 channels (RGB). The images had to be preprocessed in order to obtain the cutout of the hand and edges of the details to further proceed with training.

2. Preparing the Data

Preparing Data-Frames from the image dataset and normalizing all the values. Converting it to a numpy array and used it to fit the model.

3. Building and Training the Model

Building a CNN Model using Keras library and running 20 Epochs over the training dataset with 23k+ images.

4. Testing

Plotting a Confusion Matrix to see the accuracy of the model over the testing dataset for each alphabet with 2k+ images in total.

5. Deployment and Interface

Deploying the model over localhost by using Flask for building the backend and interface in HTML.

1. Preprocessing the Images and the Dataset

The dataset was obtained from online and had 920 images for each sign of the American Sign Language. (below - sample images from the dataset (200*200 Pixels, RBG))

F

I

Each of the images were preprocessed through various filters and operations in OpenCV python in order to obtain a grayscale image containing only the details of the edges of the hand.

```
#reading
img = cv2.imread('test\\orig_sample\\J.jpg')
#conv to grayscale
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
#noise reduction
img = cv2.GaussianBlur(img, (3, 3), 5)
```

Gaussian blur filter was used to reduce the noise from the image and Canny Edge detection was used to detect the edges from the image.

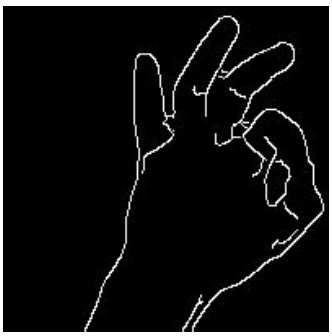
Canny edge detection compares the intensity of the neighbouring pixels to find the details and edges using the lower and higher thresholds.

After that the images were inverted and resized and appended to the dataframe.

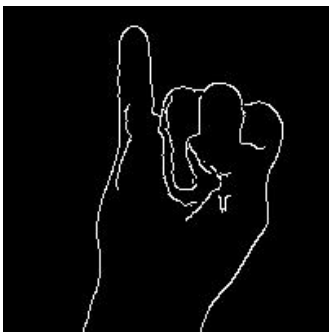
```
#edge detection
high_thresh, thresh_im = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
high_thresh2=high_thresh*0.5
low_thresh = 0.1*high_thresh
img=cv2.Canny(img,low_thresh,high_thresh)

#inverting and resizing
img =~img
img = cv2.resize(
    img,
    dsize=(100,100),
    interpolation=cv2.INTER_CUBIC
)
```

Sample images after the processing (100*100 Pixels, Single Channel, Non Inverted)



F



I

2. Preparing the Data

All the images after the preprocessing were appended to a list and the corresponding labels and their indices were appended to labels list.

The lists were further converted to a numpy array with every row representing a single image. All the rows were normalized to 1 representing the max value(White) of a pixel and 0 representing the min (Black).

```
images=np.asarray(images)
images = images.astype('float32')/255.0
```

The dataset contained 26k+ images which was split into training and testing dataset using Sklear.train_test_split() with test:train ratio being 0.05.

```
X_train, X_test, Y_train, Y_test = train_test_split(images, labels, test_size = 0.05)

X_train = X_train.reshape(X_train.shape + (1,))
X_test = X_test.reshape(X_test.shape + (1,))

label_binarizer = LabelBinarizer()
y_train_encoded = label_binarizer.fit_transform(Y_train)
y_test_encoded = label_binarizer.fit_transform(Y_test)
```

3. Building and Training the Model

Since our dataset consisted of images and we had to build a model which could learn these images well and then classify a given images to the correct label, we decided to use a Convolutional Neural Network(CNN) model for the same. The layers of a CNN consists of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layer.

The structure of our CNN Model was as shown below:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 100, 100, 1)	0
conv2d_1 (Conv2D)	(None, 98, 98, 64)	640
conv2d_2 (Conv2D)	(None, 96, 96, 32)	18464
max_pooling2d_1 (MaxPooling2	(None, 48, 48, 32)	0
dropout_1 (Dropout)	(None, 48, 48, 32)	0
conv2d_3 (Conv2D)	(None, 46, 46, 64)	18496
conv2d_4 (Conv2D)	(None, 44, 44, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 22, 22, 128)	0
dropout_2 (Dropout)	(None, 22, 22, 128)	0
conv2d_5 (Conv2D)	(None, 21, 21, 64)	32832
max_pooling2d_3 (MaxPooling2	(None, 10, 10, 64)	0
flatten_1 (Flatten)	(None, 6400)	0
dense_1 (Dense)	(None, 128)	819328
dropout_3 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 29)	3741

Total params: 967,357
Trainable params: 967,357
Non-trainable params: 0

The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing.

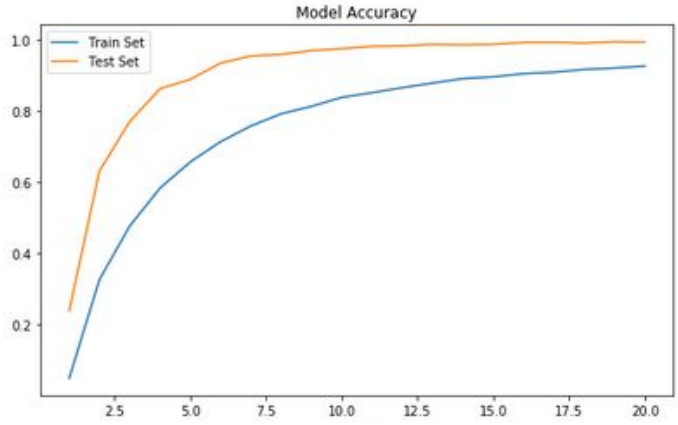
For training the model, we ran the model over the training dataset(95% of the total dataset) in 20 epochs and 64 batches and this epoch-batch combination had a very good accuracy of 90+% which is relatively good as compared to many existing models.

```
Epoch 13/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.3471
val_acc: 0.9856
Epoch 14/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.3113
val_acc: 0.9847
Epoch 15/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.2985
val_acc: 0.9861
Epoch 16/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.2714
val_acc: 0.9912
Epoch 17/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.2551
val_acc: 0.9918
Epoch 18/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.2357
val_acc: 0.9895
Epoch 19/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.2298
val_acc: 0.9933
Epoch 20/20
26661/26661 [==============================] - 30s 1ms/step - loss: 0.2121
val_acc: 0.9925
```

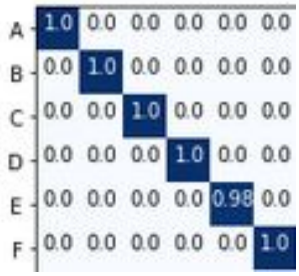
4. Testing

The dataset of 23k+ images was split to obtain a test dataset with test:train ratio of 0.05. The test dataset had approximately 1.3k images.

Below - Accuracy score of model during the 20 cycles of training (Epochs)

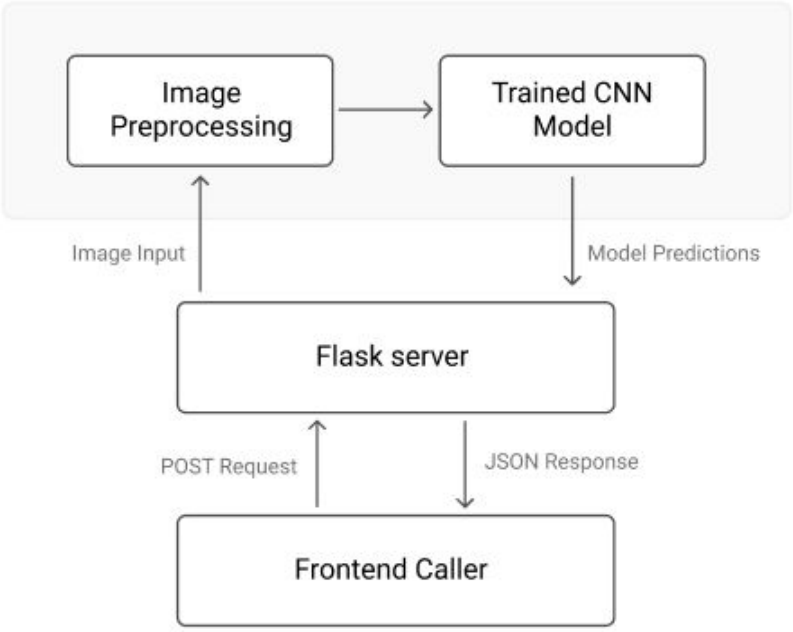


A color map was also plotted having the True labels(on Y) vs Predicted labels(on X) with accuracy score in each cell. Below is a sample of that confusion matrix.



5. Deployment

Deployment encompasses all the processes involved in getting the software up and running properly in its environment, including installation, configuration, running, testing, and making necessary changes.



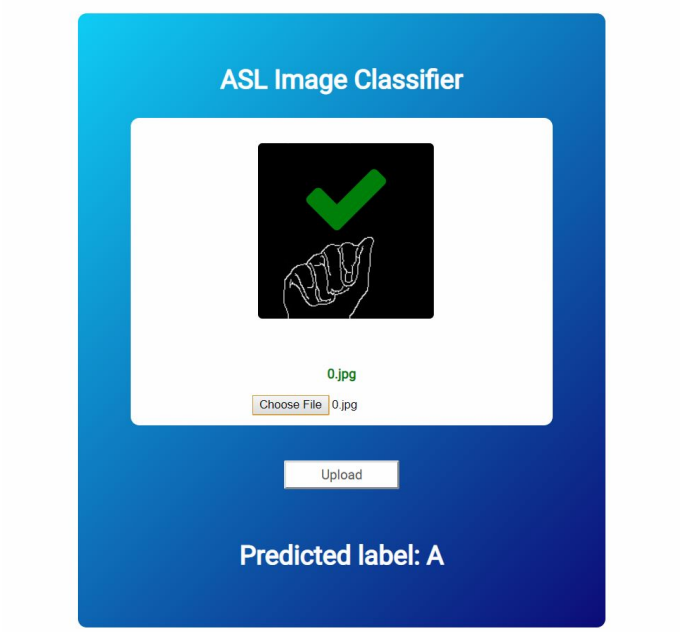
We create a URL endpoint to which anyone can make a POST request and they would get a JSON response of what the model has inferred without having to worry about its technicalities.

For this, we will create a simple Flask Server which supports uploading of an image and do some necessary image preprocessing required for our already trained model. It will then send a

response with the model inferences.

Advantages of Flask Deployment:

- When we are providing our API endpoint to frontend team we need to ensure that we don't overwhelm them with preprocessing technicalities.
- We might not always have a Python backend server (eg. Node.js server) so using numpy and keras libraries, for preprocessing, might be a pain.
- If we are planning to serve multiple models then we will have to create multiple TensorFlow Serving servers and will have to add new URLs to our frontend code. But our Flask server would keep the domain URL same and we only need to add a new route (a function).
- Providing subscription-based access, exception handling and other tasks can be carried out in the Flask app.



DISCUSSION

The project was made keeping in mind the motive to devise a method for everyone to be able to communicate using American Sign Language which would solve a lot of issues for the people who can't communicate using speech.

Though we did not plan to convert it into a full fledged application for mobile devices but learnt a lot and did solve a major problem of classifying the image input. The project has a wide variety of applications including gesture control for various devices, game controllers etc.

We achieved an accuracy of 90+%, by using a Neural Network to train our model and made an online portal to deploy the classifier using Flask framework, which uses the pre trained model and could be deployed over an online server too. We in the future hope to better the accuracy of the software, it's hand detection algorithm and convert it into an application for mobile devices.

BIBLIOGRAPHY

Al Sweigart. *Automate the Boring Stuff with Python*. 14 Apr 2015

<https://automatetheboringstuff.com/>

OpenCV Documentation <https://docs.opencv.org/3.1.0/d4>

A Comprehensive Guide To Convolutional Neural Networks | Towards Data Science

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Fingers Detection | lzane

<https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python/tree/master/material>

Flask <http://flask.pocoo.org/>