ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 10, October 2017

Convolutional Neural Networks based Sign Language Recognition

Karthick Arya[1], Jayesh Kudase [2]

Computer Engineer (B.E), Fr. Conceicao Rodrigues College of Engineering, Maharashtra, India[1]

Computer Engineer (B.E), Fr. Conceicao Rodrigues College of Engineering, Maharashtra, India[2]

ABSTRACT: A communication problem has always existed between the deaf community and the hearing majority. Recent innovations in the fields of computer vision wherein automatic sign language recognition tools have been developed has helped overcome this problem. The system proposed in this paper uses the power of Convolutional Neural Network (CNN) to recognize the hand gestures fed in through camera and predict the alphabet associated with the particular hand sign in real-time. Our system uses the pre-trained Google's Inception v3 model. We have used transfer learning technique to modify pre-trained neural network to solve the problem. The proposed system implements a predictive model built on a sparse data still giving cross validation accuracy of 100%. The system achieves an overall of 91% accuracy in real-time recognition.

KEYWORDS: Tensorflow, sign language recognition, Keras, Inception v3 model, Convolutional Neural Network (CNN)

I. INTRODUCTION

In deaf community there exists a communication barrier amongst the people and also with the outside world. To overcome this problem we have taken few Alphabets in sign language and developed a recognition system which tries to predict the sign which the person is showing in real-time. This system tries to solve the issues in sign-language communication like mistaken understanding or wrongly interpreted signs and tries to improve overall experience of the user. We have taken 5 alphabets and developed a predictive model using CNN. This model helps us to decipher the sign language in real-time and help the user understand its meaning. The alphabets chosen are: A, B, D, F, V. The hand-signs for these alphabets are as shown in Fig. 1 which were also part of the input dataset for our prediction model.

Fig.1. Hand-signs for A, B, D, E and V respectively

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 10, October 2017

II. BACKGROUND

A. *Neural Networks*

1) *Convolutional Neural Networks (CNN)*:

Convolutional Neural Networks are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs various convolutional operations. Each single image is a combination of pixels which are represented in form of a matrix. The operations are applied on this matrix. A CNN has various layers namely, convolutional, pooling, fully connected, etc. In each of these layer's unique operations are applied in the process of learning. The weights are adjusted using Error Back Propagation and finally we get a model with adapted weights which suffices our problem. Fig.2. depicts a Convolutional Neural Network.

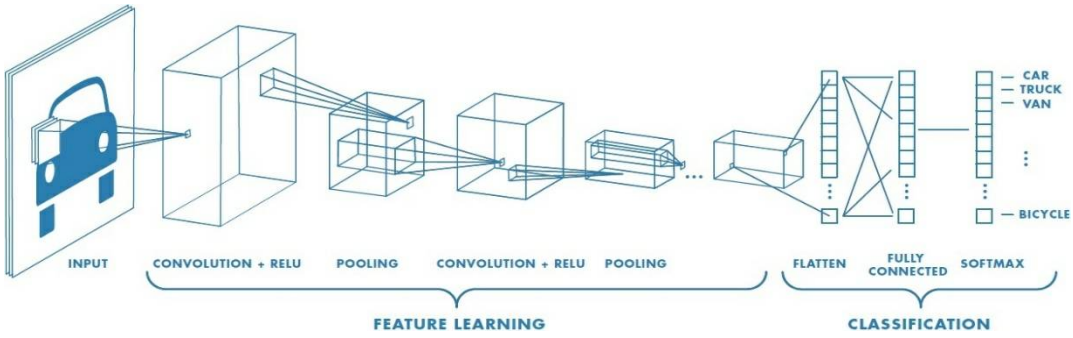


Fig.2. Convolutional Neural Network

2) *Inception v3*:

In Inception v3, the model makes the decision by doing each convolution in parallel and concatenating the resulting feature maps before going to the next layer. Each of the convolution's feature maps will be passed through the mixture of convolutions of the current layer. The idea is that we don't need to know ahead of time if it was better to do a 3×3 then a 5×5 convolution. Instead, just do all the convolutions and let the model pick what's best. Additionally, this architecture allows the model to recover both local feature via smaller convolutions and high abstracted features with larger convolutions.

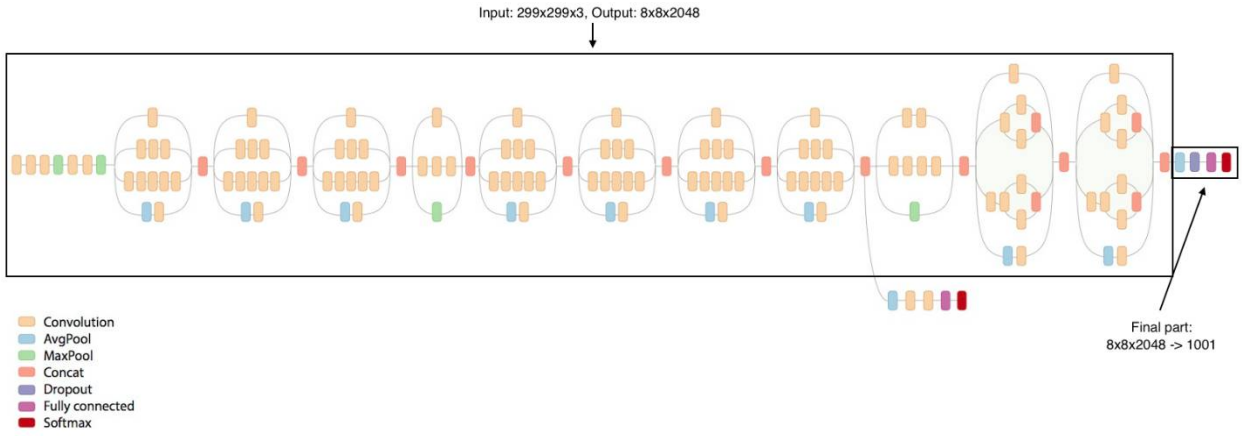


Fig.3. Inception v3 model

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 10, October 2017

B. *TensorFlow and Keras:*

1) TensorFlow:

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

2) Keras:

Keras is an open source neural network library written in Python. It is capable of running on top of MXNet, Deeplearning4j, Tensorflow, CNTK or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being minimal, modular and extensible. The library contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier.

C. *Transfer Learning:*

Modern object recognition models have millions of parameters and can take weeks to fully train. Transfer learning is a technique that shortcuts a lot of this work by taking a fully-trained model for a set of categories like ImageNet, and retrains from the existing weights for new classes. Transfer learning allows us to deal with the scenarios by leveraging the already existing labeled data of some related task or domain. The accuracies of the models which are built from scratch are also very low. Transfer-Learning helps us to leverage a pre-trained model and modify them to accommodate our requirements. These re-trained models are optimized, have high accuracy and it is easy for us to adapt this CNN to solve our problem.

In our case, we have used pre-trained model of Google InceptionV3CNN and modified it. We basically replace the topmost (output) layer with two other layers and train the model.

1) *Modify final layer of pre-trained model:*

A pre-trained model for feature extraction is used in this step. The output layer (the one which gives the probabilities for being in each of the 1000 classes) is removed and then the entire network is used as a fixed feature extractor for the new data set. Following is a code snippet that we used in our project for performing this step.

```
# create the base pre-trained model and remove output layer.
base_model =InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x =GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x =Dense(1024, activation='relu')(x)
# and a logistic layer -- let's say we have 200 classes
predictions =Dense(200, activation='softmax')(x)

# this is the model we will train
model =Model(inputs=base_model.input, outputs=predictions)
```

2) *Use the Architecture of the pre-trained model:*

In this step we can use architecture of the model while we initialize all the weights randomly and train the model according to our dataset again.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 10, October 2017

3) *Train some layers while freeze others:*

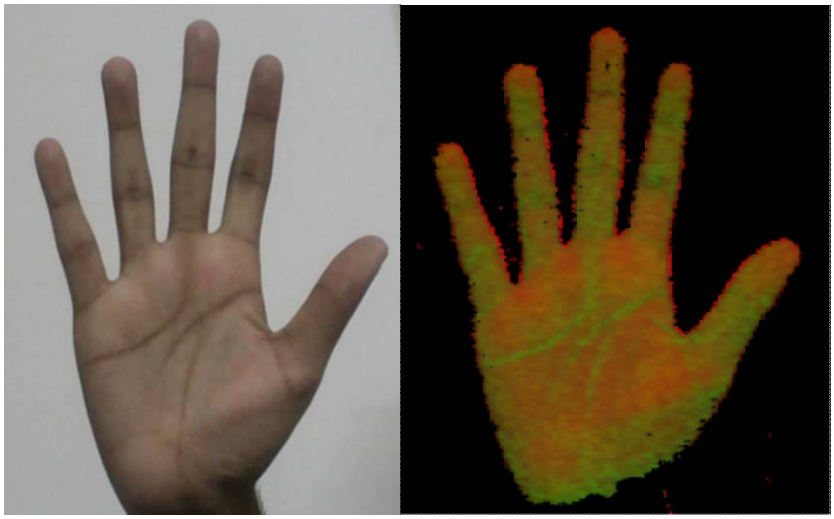
After that we use a pre-trained model to train partially. The weights of initial layers of the model frozen while only the higher layers are retrained. Following is a code snippet that we used in our project for performing this step.

```
# we chose to train the top 2 inception blocks, i.e. we will freeze the first 249
layers and unfreeze the rest:
for layer in model.layers[:249]:
   layer.trainable =False
for layer in model.layers[249:]:
   layer.trainable =True
```

III. METHODOLOGY

A. *Data Collection:*

The training data for the problem is collected through web camera of the laptop. For each sign/label, 50 samples of raw images from web camera is collected. We have not stored this input data in raw format. For the sake of improvement in accuracies and avoiding unwanted data, we have converted the images into HSV format and performed masking and bitwise operations to identify just human color out of the image. Example of a raw input image and its converted HSV image is shown in Fig.4.



Raw image from webcam Converted HSV image

Fig.4. example of a raw and its converted HSV image

B. *Data Transformation:*

50 samples per label/sign is an insufficient data. In order to build a powerful image classifier, we have to make the most out of the small dataset. In order to overcome this problem, we augmented the images from our dataset via a number of random transformations. We generated 10 transformed images for each of the input sample image we captured by applying operations like rotation, scaling, flip, zooming, etc. Following Fig.5. is an example of how the images look after transformations are applied on the converted HSV image from Fig.4.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 10, October 2017

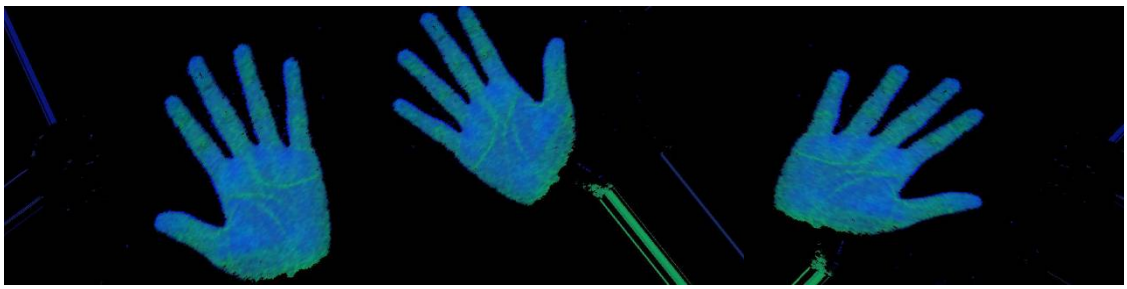


Fig 5. Transformed images

C. *Training:*

After performing data transformation for each label, the size of sample is now 550. As we are using 5 labels, therefor total size of sample data for training = 550x5 = 2750. (n=2750). Once training data is available, we trained the modified InceptionV3 CNN with shuffled data for 10 epochs with training set = 90% (n=2475) and validation set = 10% (n=275).

D. *Fine-Tuning:*

Once whole CNN is trained for 10 epochs, we have implemented fine-tuning technique by freezing the top 249 layers of CNN and training just the top two layers for improved accuracy as mentioned in third step of transfer learning.

IV. RESULTS

A. *Training Accuracy:*

The CNN model trained for the obtained dataset had initial Training set and validation set accuracy of 72.8.5% and 43.2% respectively in epoch-1 and the final training set accuracy and validation set accuracy of 99.5% and 100% respectively in epoch-10.

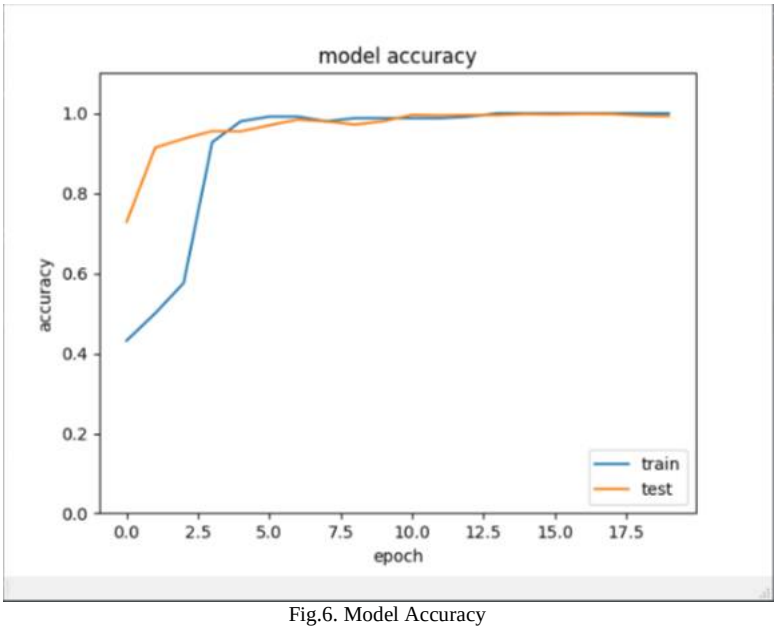


Fig.6. Model Accuracy

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 10, October 2017

B. *Training Loss:*

The trained CNN model had Initial training and validation loss of 1.62 and 6.41 respectively in epoch-1 and final training and validation loss of 0.0084 and 0.0051 respectively at end of epoch-10.

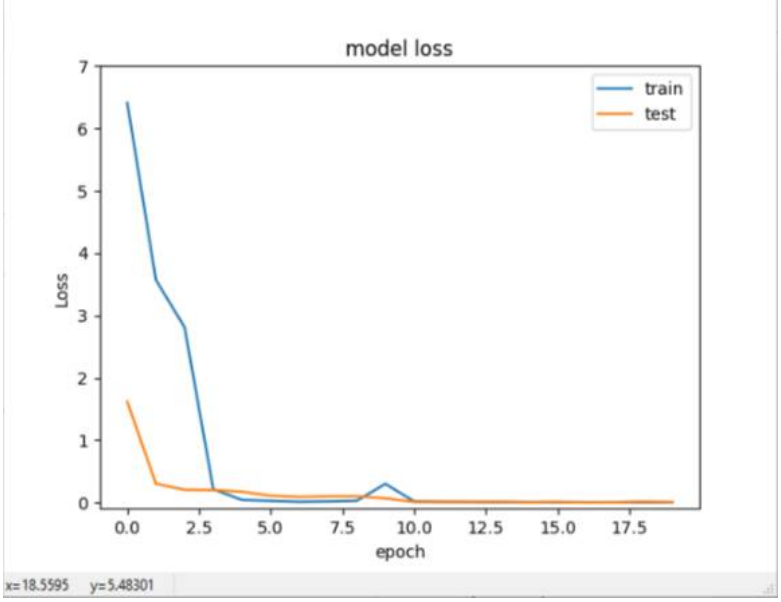


Fig.7. Model loss

C. *Test Accuracy::*

The test accuracy of each sign for each alphabet in real-time feed through webcam is given in the table below:

Sign	Accuracy
A	91%
B	95%
D	92%
F	89%
V	90%

Fig. 8. Test accuracy table

V. FUTURE SCOPE

The predictive model proposed in this paper is trained for recognition of only 5 characters. The model can be refined in the future and can be trained to recognize all the 26 alphabets from the sign language. The input sample for building the model was also sparse. An input with a lot of variations and large number of sample images can be effectively used to train the proposed model in order to increase its efficiency and accuracy as well. Further the model can be developed to process the continuous input stream of sign language and convert them into their corresponding sentences.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 10, October 2017

REFERENCES

1. ProBlogger Contributors, Available at https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
2. Keras Documentation, Available at https://keras.io/applications/#inceptionv3
3. OpenCV-Python Tutorials contributors, Available at http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html
4. Tommy Mulc, Inception modules: explained and implemented, Available at https://hacktilldawn.com/2016/09/25/inception-modules-explained-and-implemented/
5. TensorFlow Contributors, Available at https://www.tensorflow.org/install/install_windows#determine_which_tensorflow_to_install
6. Nitish S. Mutha, Install TensorFlow with GPU for Windows 10, Available at http://blog.nitishmutha.com/tensorflow/2017/01/22/TensorFlow-with-gpu-for-windows.html
7. Google Research blog Contributors, Available at https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html