Asian Journal of Convergence in Technology Volume IV Issue I
ISSN NO: 2350-1146 I.F-5.11

Automatic ASL Gesture Recognition System Using Convolutional Neural Network

A supervised Feature Learning Approach

Pooja Jaiswal, CSE
IIIT-Naya Raipur

Pooja Sonkar, CSE
IIIT-Naya Raipur

Nitesh Agarwal, CSE
IIIT-Naya Raipur

***Abstract* –American Sign Language Gesture Recognition project aims to develop a model which can recognize different sign language gestures used by people who are deaf or hard of hearing and also who are able to hear them but cannot physically speak. It is based on Human-Machine Interaction, which is a broad research field with application in Robotics, Gaming, and Home Automation etc. This system helps people to understand the sign language and makes the communication with deaf and dumb people easier. The proposed model uses concepts of deep learning, specifically Convolution Neural Network for training and testing the model.**

Keywords: Deep Learning, Computer Vision, Convolution Neural Network, Feature Extraction.

I. Introduction and Motivation

The most commonly used sign language that serves the deaf-communities in the United States of America is the natural language called American Sign Language (ASL). Deaf and dumb people use sign language for their communication but the main problem arises when normal people are not able to understand what they want to express. The idea of recognizing sign language is an interesting machine-learning problem while simultaneously being very useful for deaf people to communicate with people who don't understand American Sign Language (ASL).(Example of ASL shown in Fig: 1)

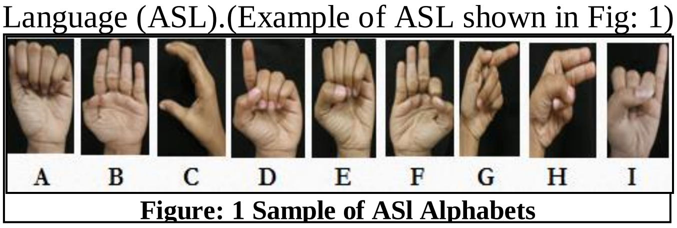


Figure: 1 Sample of ASl Alphabets

The solution to this problem is proposed in this paper using the Supervised Feature Learning concept of Convolutional Neural Networks (CNN) for classifying images of sign language taken as input and passing the feature vector of the input images through channels of neural networks. CNNs are useful to automate the process of feature extraction/construction and classify the image inputs.

A more analyzed overview of what CNNs do would be like assume that you take the image, pass it through a series of convolutional, nonlinear, pooling (down-sampling), and fully connected layers, and get an output. As we said earlier, the output can be a single class or a probability of classes that best describes the image. So basically we are imitating a neuron and trying to make it learn the art of recognition and classification.

II. Related Work

The research of Sign Language gesture recognition has gained much attention because of its applications for the interactive human-machine interface. Recognition process of any hand gesture can be done through two different approach first gloves based and other is the computer-based approach. Glove based techniques increase complexity hence computer vision method is preferred over it. Computer vision is categorized in further two ways 2D/3D arm model and on its appearance based. Appearance-based gesture recognition is used specifically for communication gesture. Appearance-based approaches have several advantages including low computational complexity, real-time processing, and so on. Most researchers, therefore, adopt appearance-based approaches[3].

In [4] the authors have successfully classified up to 92−93% of the letter using a linear or Gaussian kernel SVM, outperforming k-nearest neighbor classification by about 10.But SVM being binary classifier is less efficient in case of a large number of classes.

Instead of constructing complex handcrafted features, CNNs are able to automate the process of feature construction. This model is able to recognize gestures with high accuracy. The predictive model is able to generalize on users and surroundings not occurring during training with a cross-validation accuracy of 91.7% [5]. The success of CNNs partly lies in its invariance to translation, rotation and scale, which is also due to its ability to learn high level semantically.[6]. In [7]they have used an ensemble approach to classify images i.e. using Random forest Algorithm for identifying 24 alphabets of American Sign Language. A detailed study of [8] gave us an idea that K-nearest neighbor algorithm can also be used to classify images.

III. PROBLEM FORMULATION

The main problem that is to be dealt in this project is to develop a model for recognizing American Sign Language used by people who cannot hear or speak. The first step after converting data into the suitable format is to extract image features, mainly pixel arrays of the images in datasets. This will guide our machine in distinguishing different Alphabets of American Sign Language.

Input datasets use static images of hand gestures and at initial level focus will be on alphabets whose gestures are static. In this context, letter ‘R’ and ‘Z’ will require video frames, but our datasets include only static images. All the input datasets are classified by using convolutional neural network classifier.

IV. DATA COLLECTION

Sign Language recognition systems are tested with a very large, complete, standardized and intuitive database of sign language. In the area of information science, the most important task is to find predictive relationships from data.

A data set is divided into 3 categories.

A. *Training set:* A set of data points used to make the model learn, i.e. to fit the parameters in the classifier.

B. *Validation set:* A set of data points used to tune the hyper parameters of a model.

C. Test set: A set of data points used only to assess the performance [generalization] of a fully-specified model.

A data set of 7500 images is collected. This dataset consists of 24 English alphabets gestures in ASL (American Sign Language). This data set doesn’t include letter “R”,”Z”. 80% of the data set i.e. 6000 images are kept as the training set. And the remaining 20% of the data set i.e. 1500 images are used for validation. Test set will be approximately 20% of the total data set i.e. 1500 images. This will be created by us. This step is taken to undertake robustness check of the project and make it real-time. The users and backgrounds are not contained in the training set and Validation set. The Training set and validation set has been segmented to contain only the part where the palm is recognized. Else everywhere the image contains black shade.

V. DATA PREPROCESSING

To transform raw data into a machine-understandable format we are required to apply mining techniques that involve Data Pre-processing.

The method of dividing an image into multiple parts is called Image segmentation. This is majorly used to identify objects or other relevant information in digital images. The dataset used here contains already segmented images, foreground extraction was used to segment the palm region from the original image.

First, all the images in the dataset are converted to gray-scale. Since our system needs to be lighting invariant, the colour information should actually be ignored because we cannot rely on the coloration of the hand to be consistent between test and training image data. To convert the images to the greyscale cv2 module of python was implemented.

Second, all the images in the dataset need to be of a common size so that while creating a feature vector the size of the vector is common for all. The image size was converted to 200*200 pixels.

Our final dataset consists of hand gestures for 24 letters of the ASL alphabet bounded in a 200x200 bounding box, as shown in Fig 4. We have 100-400 samples for each letter, giving us a total size of 7500 images for our entire dataset of 24 letters with the total of 40,000 greyscale-converted pixels in each image. The pixels from each image is extracted and stored into a numpy array file.(see figure 2)

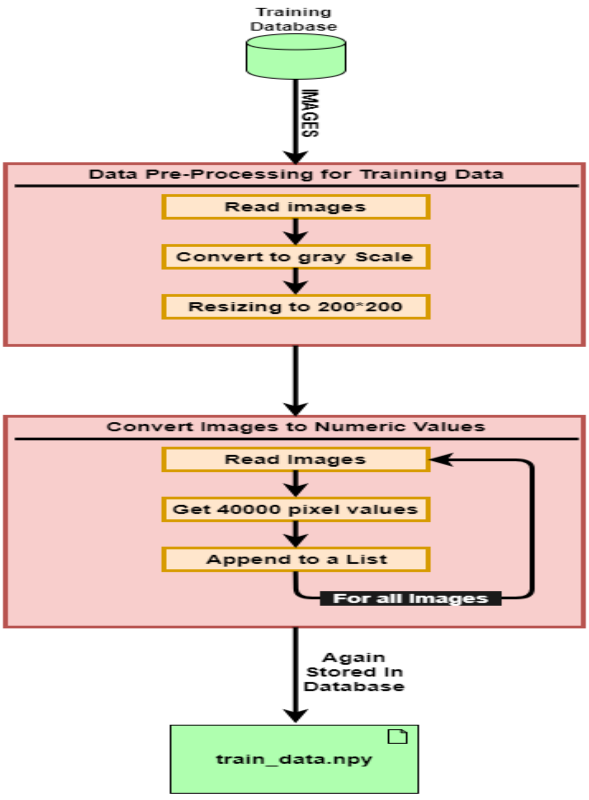


Figure: 2 Data Pre-processing

VI. NETWORK ARCHITECTURE

The two most common and effective approach to make the model learn how to recognize different sign languages according to literature survey is using Convolutional Neural Network.

When a computer sees an image (takes an image as input), it will see an array of pixel values. Depending on the resolution and size of the image in our dataset, it will see a 200 x 200 array of numbers. We have greyscale images in JPG format. Each of these numbers is given a value from 0 to 255 which describes the pixel intensity at that point. These numbers, while meaningless to us when we perform image classification, are the only inputs available to the computer. The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class.

When the array of numbers i.e. the feature vector is ready, we pass the training dataset to the layers of Convolutional Neural Network so that the model learns to recognize the 24 classes of hand gestures. The Network consists of multiple blocks which include convolution layer(ReLU activated) and max-pooling layer followed by a fully connected dense layer. The model is trained over the training dataset and validated with the validation set after each epoch to get an idea of how accuracy increases or decreases(see figure 3).

VII. PERFORMANCE ACCELERATION

Deep learning using Convolutional Neural Network involves a lot of computational tasks which in turn consumes a lot of time. So, to accelerate the processing one can use GPU (Graphics User Interface). CUDA is a well known platform for parallel computing and programming model which was developed by NVIDIA for general computing on (GPUs) graphical processing units. With CUDA, developers are able to speed up computing applications dramatically by harnessing the power of GPUs. The CUDA (Toolkit) from NVIDIA provides everything that is needed to develop applications that are GPU-accelerated. The CUDA Toolkit includes a compiler, few GPU-accelerated libraries, tools for development and the CUDA runtime.

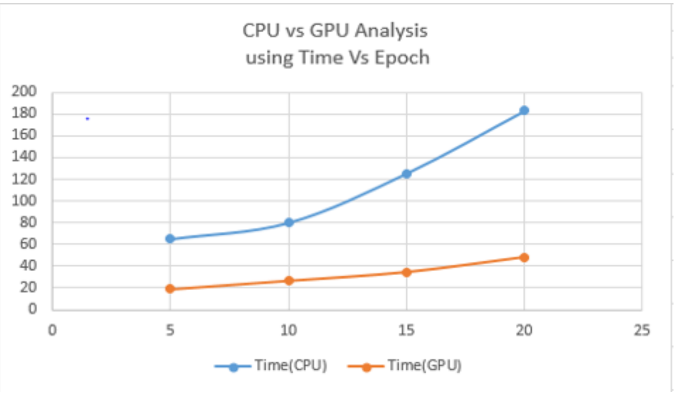


Figure: 5 CPU-GPU Analysis

cuDNN is a GPU accelerated library for the primitives of Deep learning Neural Networks. It consists of efficient context-based APIs that allows easy interpretability. The proposed model uses tensorflow from the cuDNN library of the CUDA toolkit to train the model.

Tensorflow that can be controlled by a simple python API is a framework to perform computation very efficiently and it can also tap into the GPU to accelerate the performance of the model even further.

The time vs epochs Graph is used to show the drastic change in performance of the model while using GPU and while using CPU.

VIII. RESULT

In Convolution Neural Network, the input stream is passed to each layer called Feed Forward step. Based on output‘s loss, these data are propagated backward for readjustment of weights. This step is called backpropagation. During this training period, one forward pass and one backward pass of training data are called one epoch. After each epoch weights converges towards the accurate value resulting in validation accuracy. Hence increasing no. of epochs will increase the accuracy of the dataset.(Figure 7).

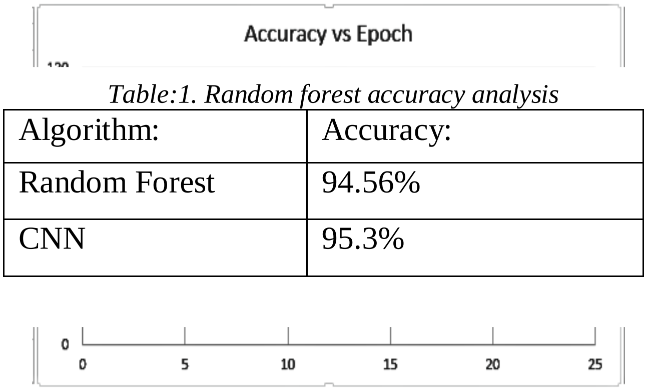


Table:1. Random forest accuracy analysis

Algorithm:	Accuracy:
Random Forest	94.56%
CNN	95.3%

Figure: 6 Accuracy Analysis

Training the model for 10- 15 epochs gives maximum accuracy after which graphs became stable.

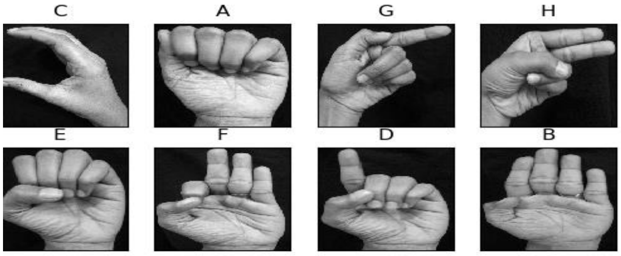
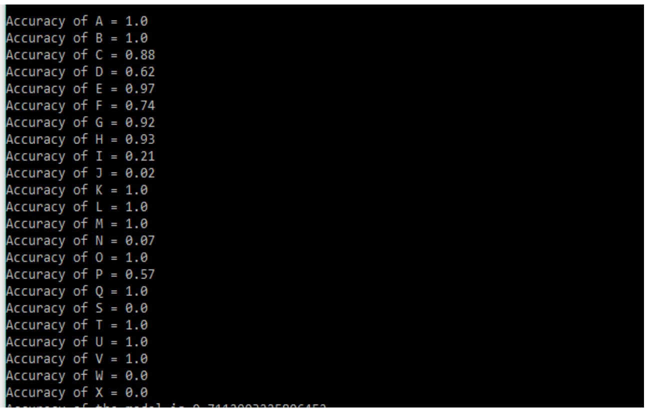


Figure: 7 Glimpse of final output

Hence after training the model for 15 epochs, we got an accuracy of 95.3% with a loss of 4.7% on the validation set. (Figure 4)

Comparing each alphabet individually our trained model predicted them with an accuracy as shown in Figure 5.



```
Accuracy of A = 1.0
Accuracy of B = 1.0
Accuracy of C = 0.88
Accuracy of D = 0.62
Accuracy of E = 0.97
Accuracy of F = 0.74
Accuracy of G = 0.92
Accuracy of H = 0.93
Accuracy of I = 0.21
Accuracy of J = 0.02
Accuracy of K = 1.0
Accuracy of L = 1.0
Accuracy of M = 1.0
Accuracy of N = 0.07
Accuracy of O = 1.0
Accuracy of P = 0.57
Accuracy of Q = 1.0
Accuracy of S = 0.0
Accuracy of T = 1.0
Accuracy of U = 1.0
Accuracy of V = 1.0
Accuracy of W = 0.0
Accuracy of X = 0.0
```

Figure: 8 Individual Alphabet analysis

Table:3. SVM accuracy analysis.

ALGORITHM	ACCURACY
SVM	89%
CNN	95.3%

Due to the strong similarity in a gesture of few alphabets the accuracy of that alphabets is degraded.

IX. COMPARISON WITH OTHER ALGORITHMS

Random Forest*:* CNN and Random forest are comparable because both use a type of divide and conquer approach in one way or the other. As per our literature survey, Random forest is also suitable for American Sign Language Recognition[7]. So we tested the algorithm of random forest on our own dataset and compared the efficiency parameters with CNN.

K-Nearest Neighbour*:* It is a non parametric method which is used for classification/regression. In K-NN, the input consists of the k-closest training examples from the extracted feature space.

Table:2. KNN execution time analysis

Algorithm:	Execution Time:
K-NN	>35 minutes
CNN	5.56 minutes(5 epochs)

K-NN is a Lazy classifier[8]. K-NN cannot be used for classifying ASL because it is extremely slow as it compares one data to the whole training set data and sort it according to the distance which takes a lot of computational time.

Support Vector Machine*:* Support Vector Machine is known be a kernel-based classifier. These are the group of supervised machine learning algorithm that can be used for classification. It is a binary classifier. One-v/s-rest approach is used to implement this algorithm for multiclass classification. Given M classes in a problem, we need to independently train M linear SVMs, and the data belonging to the other classes are considered as negative cases during the training process.
When we have too many classes that many numbers of classifier need to be trained. This is the reason SVM proves to be a not-so-efficient algorithm for large datasets.

X. CONCLUSION AND FUTURE SCOPE

The current model used for classifying input images based on CNN gives 95.3% accuracy on the validation set. Excluding very similar classes, average accuracy for every alphabet came greater than 90%.

In future, our aim will be to cover non static alphabet's gesture too. With increasing speed, the indistinctive gesture will be tried to classify at real time through video analysis more accurately by using R-CNN [9] and SVM combined with CNN, which are advanced deep learning algorithm. The main aim will also be to increase test data accuracy.

XI. ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our mentor Dr. Muneendra Ojha Sir, for helping us throughout this project and guiding us at every step.

XII. REFERENCES

[1] E. Shelhamer, J. Long and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 640-651, April 1, 2017

[2] M. Blot, M. Cord, and N. Thome, "Max-min convolutional neural networks for image classification," 2016 IEEE Int. Conf. Image Process., pp. 3678–3682, 2016.

[3] D. K. Ghosh and S. Ari, "Static Hand Gesture Recognition Using Mixture of Features and SVM Classifier," 2015 Fifth Int. Conf. Commun. Syst. Netw. Technol., pp. 1094–1099, 2015.

[4] M. Hasan, T. H. Sajib, and M. Dey, "A machine learning based approach for the detection and recognition of Bangla sign language," 2016 Int. Conf. Med. Eng. Heal. Informatics Technol., pp. 1–5, 2016.

[5] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign Language Recognition Using Convolutional Neural Networks," Eur. Conf. Comput. Vis., pp. 572–578, 2015.

[6] X. Yingxin, "A Robust Hand Gesture Recognition Method Via Convolutional Neural Network," 2016 6th Int. Conf. Digit. Home, pp. 1–4, 2016.

[7] C. Dong, M. C. Leu, and Z. Yin, "American Sign Language alphabet recognition using Microsoft Kinect," 2015 IEEE Conf. Comput. Vis. Pattern Recognit. Work., pp. 44–52, 2015.

[8] D. Aryanie and Y. Heryadi, "American sign language-based finger-spelling recognition using k-Nearest Neighbors classifier," 2015 3rd Int. Conf. Inf. Commun. Technol. ICoICT 2015, pp. 533–536, 2015.

[9] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, June 1 2017.