International Research Journal of Engineering and Management Studies (IRJEMS)
Volume: 03 Issue: 03 | March -2019 e-ISSN: 2395-0126

ASL DETECTION WITH DEEP LEARNING FRAMEWORK

Puvvula Lakshmi Mounika[1], Oggu Valli Padmam[2], Tubati Mounica[3], Pittala Sai Sowmya[4]

Vasireddy Venkatadri Institute of Technology, Nambur, Guntur District, Andhra Pradesh

---***---

Abstract – *Automatic Sign Language is the important communication bridge between deaf individuals, and it is also used by hard-of-hearing individuals. The language employs signs made with the hands. The implementation of an Automactic Sign Language (ASL) is based on convolutional neural network which is a deep learning framework. The concept behind the project is Image Classification which is a feature extraction and pattern matching technique to classify various objects. We employ the feature extraction algorithm to characterize features and recognize symbols by comparing them with the already stored trained data set. The algorithm device is capable of extracting signs from video sequences under minimally cluttered and dynamic background. Google's Tensor flow library with keras framework is the important feature of this project. Experimental results shows satisfactory outputs for testing data for both accuracy and loss functions.*

Key Words: Automatic Sign Language (ASL), Image Classification, Convolutional Neural Network, Keras, Tensorflow, OpenCV and Spyder, an open source environment in Anaconda.

1. INTRODUCTION

Sign language is a unique type of communication that often goes understudied. It is a novel system to aid in communicating with those having vocal and hearing disabilities. This paper discusses an improved method for sign language recognition, the translation process between signs and spoken or written language. In our research, we look at American Sign Language (ASL), which is a natural language that serves as the predominant sign language of deaf communities in the United States and most of Anglophone Canada. There are 26 hand shapes that correspond to the 26 letters of the alphabets.

Communication is one of the basic requirements for survival in society. Deaf and dumb people communicate among themselves using sign language but normal people find it difficult to understand their language. Extensive work has been done on American Sign Language which uses single hand for communicating. Our project aims at taking the basic step in bridging the communication gap between normal people and deaf and dumb people. There are two main steps in building this model. The first step is to extract features from the input images. This will result in a representation consisting of one or more feature vectors, also called as descriptors. This representation will aid the computer to distinguish between the possible classes of actions. The second step is the classification of the action. A classifier will use these representations to discriminate between the different signs. In our work, the feature extraction is automated by using convolutional neural networks. An artificial neural network is used for classification.

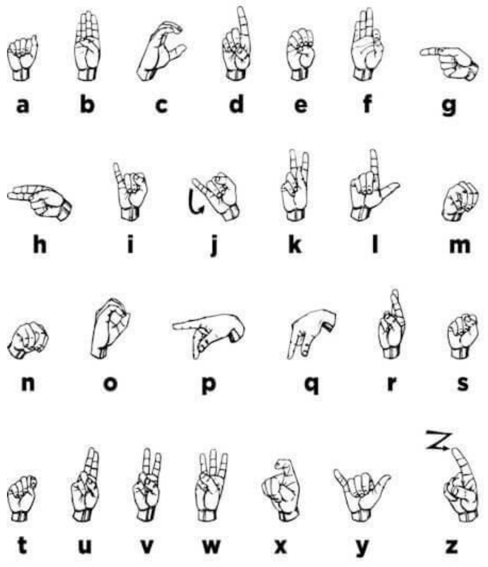


Figure 1.American Sign Language Alphabet

2 RELATED WORKS

Great improvements have been done previously on Automatic Sign Language. One of the approaches included key point detection of Image using SIFT and then matching the key point of a new image with the key points of standard images per alphabet in a database to classify the new image with the label of one with the closest match[1]. Another one calculated the eigen vectors of covariance matrix calculated from the vector representation of image and used Euclidean distance of new image eigen vector with those in training data set to classify new image[3]. Some of them used Neural networks for training but their dataset comprised of only single banded images and their feature vectors are based on the angle between fingers, number of fingers etc[4]. The major problem with all these works that there was no mention of where the dataset was collected from and from the images it appeared as if it was taken from webcam by the people working themselves. IN a thesis for related work we found at [8], the authors had formed the dataset on one of their team members and divided that into training and testing sets before reporting the accuracy. Also they had tested only on a subset of the English alphabets. Here dataset collection part from just one person is not very appealing and hence our team decided to go for mask approach where any persons hand can be recognized by the model with a very limited dataset formed by ourselves.

Another observation is taken from the UdacityMLND_Capstone proposal where they have collected a 2GB dataset for training the model. The dataset comprised of different angles of a single hand symbol. Even though there is a change in the background cluster, their model exactly detects the sign [8]. Hence we followed the model but created the mask dataset.

3 METHODOLOGIES

Convolutional Neural Networks have been extremely successful in image recognition and classification problems, and have been successfully implemented for human gesture recognition in recent years. In particular, there has been work done in the realm of sign language recognition using deep CNNs, with input-recognition that is sensitive to more than just pixels of the images. With the use of cameras that sense depth contour, the process is made much easier via developing characteristics depth and motion profiles for each sign language gesture.

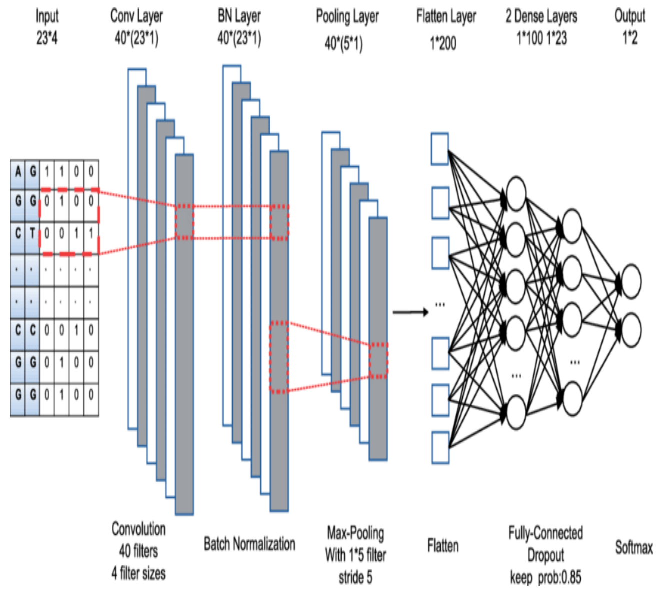
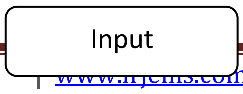

Input 23*4
Conv Layer 40*(23*1)
BN Layer 40*(23*1)
Pooling Layer 40*(5*1)
Flatten Layer 1*200
2 Dense Layers 1*100 1*23
Output 1*2
Convolution 40 filters 4 filter sizes
Batch Normalization
Max-Pooling With 1*5 filter stride 5
Flatten
Fully-Connected Dropout keep prob:0.85
Softmax

Figure 2.Basic CNN model

3.1 Architecture

Most implementations surrounding this task have attempted it via transfer learning, but our network was trained from scratch. Our general architecture was a fairly common CNN architecture, consisting of multiple convolutional, pooling and dense layers. The architecture included three groups of one convolutional layers followed by a maxpooling layer, and then flattening is done on the network followed by one fully connected layer with dense layer of relu activation function which is followed by a dropout layer followed by dense layer of softmax function and one final output layer. The following figure is our network architecture used for our model.



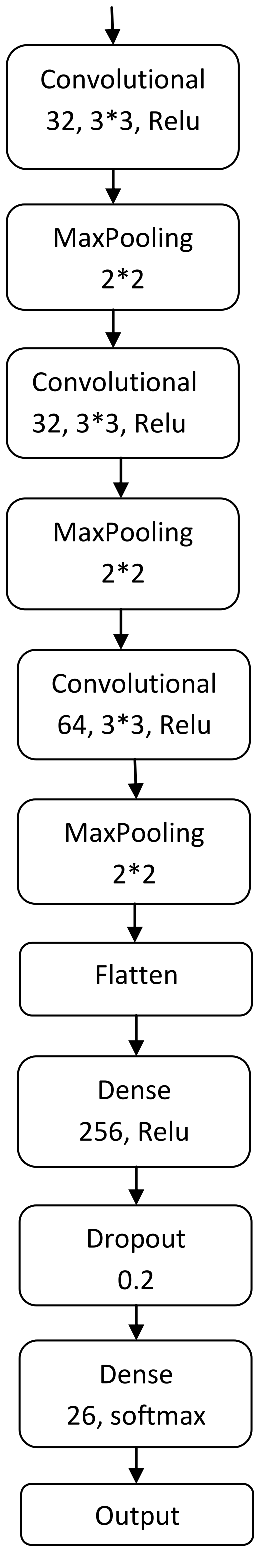


Figure 3.Network Architecture

Convolutional layer consists of a set of filters. The filters take a subset of input data at a time, but are applied across the full input (by sweeping over the input). The operations performed by this layer are still linear/matrix multiplications, but they go through an activation function at the output, which is used usually a non linear operation. We utilize the fact that consecutive layers of the network are activated by "higher" or more complex features that are executed by a large area of the networks input data. A pooling layer effectively down samples the output of the prior layer, reducing the number of operations required for all the following layers, but still passing on the valid information from the previous layer. A dense layer or fully connected layer is a linear operation in which every input is connected to every output by a weight (so there are N inputs *N output weights-which can be a lot!). Generally followed by a in non linear activation function. In our model the non linear function is ReLU.

3.2 Training Method

The convolutional layers extracts the features from the input images of size (64 64 3).The flattening step is needed so that you can make use of fully connected layers after some convolutional layers (which only observe some local part of an image by using convolutional filters). This means you can combine all the found local features of the previous convolutional layers. Each feature map channel in the output of a CNN layer is a "flattend" 2D array created by adding the results of multiple 2D kernels(one for each channel in the input layer).

3.2.1 Regularisation

Regularisation techniques like dropout which refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. By "ignoring", I mean these units are not considered during a particular forward or backward pass. More technically, at each training stage, individual nodes are either dropped out of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. The dropout rate for this model is 0.5.

Another regularization technique is data augmentation is a way of creating new data with different orientations. The benefits of these are two folds, the first being the ability to generate more data from limited and secondly it prevents over fitting. Working with limited data has its own challenges, using data augmentation have positive results only if the augmentation techniques such as flipping the images upside down orientation. However if the dataset consists of images of prescription medications then it makes sense to have a number of orientations as these images could theoretically be in any orientation. There can be many variations that can affect the results such as the size of the dataset, augmentation techniques, batch size, image size and training parameters to name a few.

3.2.2. Activation Function

The activation function used is **Rectified Linear Units (ReLUs)** for the hidden layers. A rectified linear unit has output 0 if the input is less than 0 , and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. ReLU's machinery is more like a real neuron in your body.

$$F(x) = \max(x, 0)$$

ReLU activations are the simplest non-linear activation function you can use, obviously. When you get the input is positive, the derivative is just 2, so there isn't the squeezing effect you meet on back propagated errors form the sigmoid function, research has shown that ReLUs result I much faster training for large networks. Most frameworks like TensorFlow and TFLearn make it simple to use ReLUs on the hidden layers, so you won't need to implement them yourself.

The rectifier is an activation function defined as the positive part of its argument:

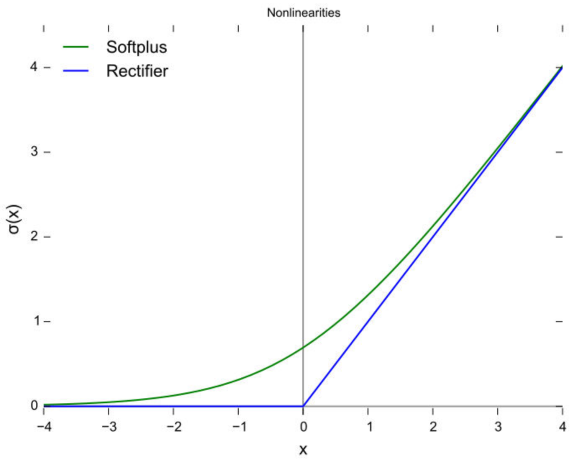


Figure Rectifier and softplus functions near x=0

$$F(x) = x^{+} = \max(0,x)$$

Where x is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification. This activation function finds applications in computer vision and speech recognition using deep neural nets.
A smooth approximation to the rectifier is the analytic function

$$F(x) = \log(1+e^{x}),$$

Which is called the softplus or SmoothReLu function. The derivate of softplus is
$F'(x) = e^{x}/1+e^{x} = 1/1+e^{-x}$, the logistic function. The logistic function is a smooth approximation of the derivative of the rectifier, the Heavy side step function. There are different types of ReLU functions like Noisy ReLUs, Leaky ReLUs, Parmetric ReLUs and ELUs.

Softmax function takes an n-dimensional vector of real numbers and transforms it into a vector of real number in range (0,1) which add upto. As the name suggests, softmax function is a "soft" version of max function. Instead of selecting one max value, it breaks the whole(1) with maximal element getting the largest position of the distribution, but other smaller elements getting some of it as well. This property of softmax function that it outputs a probability distribution makes it suitable for probabilistic interpretation in classification tasks.
Cross entropy indicates the distance between what the model believes the output distribution should be, and what the original distribution really is,

$$H(y,p) = -\sum_{i} y_{i} \log(p_{i})$$

Cross entropy measure is a widely used alternative of squared error. It is used when node activations can be understood as representing the probability that each hypothesis might be true, i.e., when the output is a probability distribution. Thus it is used as a loss function in neural networks which have softmax activations in the output layer.

3.2.3 Loss Function

The loss function in our model is categorical cross-entropy which is used in multi-class classification. Multi-class classification means having more than two exclusive targets. The targets should be encoded as one hot vectors. When using the categoraical cross entropy loss, the target should be in the categorical format(example: if you have 10 classes, the target for each sample should be a 10D vector i.e., all zeros expect for a 1 at the index corresponding to the class of the sample).

$$\prod_{c=1}^{C} y_c(\mathbf{x}, \mathbf{w}_c)^{t_c}$$

3.2.4 Optimization function

Stochastic gradient descent (SGD), also known as incremental gradient descent, is an iterative method for optimizing differentiable objective function, a stochastic approximation of gradient descent optimization. It is called Stochastic because samples are selected randomly (or shuffled) instead of as a single group or in the order they appear in the training set. The SGD is also known as on-line gradient descent. This is relatively less common to see because in practice due to vectorized code optimizations it can be computationally much more efficient to evaluate the gradient for 100 examples, than the gradient for 1 example 100 times. Eventhough SGD technically refers using a single example at a time to evaluate the gradient, you will hear people use the term SGD even when referring to mini-batch gradient descent.

3.2.5 HSV model

During training, dropout and data augmentation are sued as main approaches to reduce over fitting. The data augmentation is performed in real time on the CPU during the training phase while the model
Unlike RGB model, HSV model separates the color and intensity components which makes it more robust to lighting and illumination changes. So in this approach, we transformed the image from RGB space to HSV space. We then retain the pixels having H and S values in the range 25<H<230 and 25<S<230, to retain skin pixels and discard background. But the HSV model captures a lot of noise. Hence we can improve the performance by changing the range through bars.
Experiments are conducted on one machine with a quad-core processor (Intel Core i5), 8GB RAM. The models are implemented using the Python libraries like Tensor flow with keras, numpy, matplotlib and OpenCV for computer video detection.

4. RESULT

We initially trained and tested on a self-generated dataset of images we took ourselves. This dataset was a collection of nearly 45,000 images. Our dataset is divided into training and testing sets. For training dataset, nearly 40,000 images with 1500 images for each alphabet. For testing dataset, nearly 5,000 images with 225 images for each alphabet. Since images dataset was not constructed in a controlled setting, it was especially prone to differences in light, noise and other differences in the environment that the images were captured in. Additionally, a pipeline was developed that can be used so people are able to generate and continue adding images to this dataset. We initially split the dataset into two for training and testing, the model accuracy showed to be high and the loss is very low. Training the network on each item of the set once is an Epoch and we have trained the model for 25 Epochs. The result for both model accuracy and model loss is shown below.

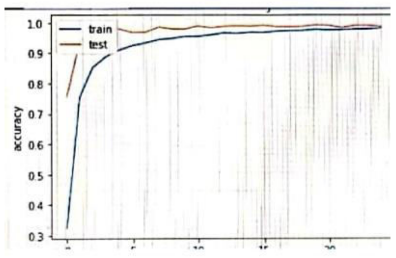


Figure 4.Model Accuracy

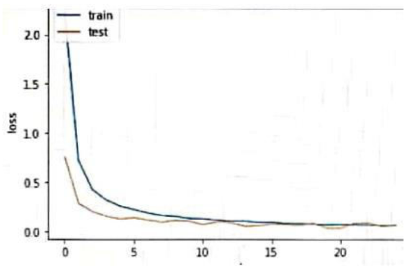


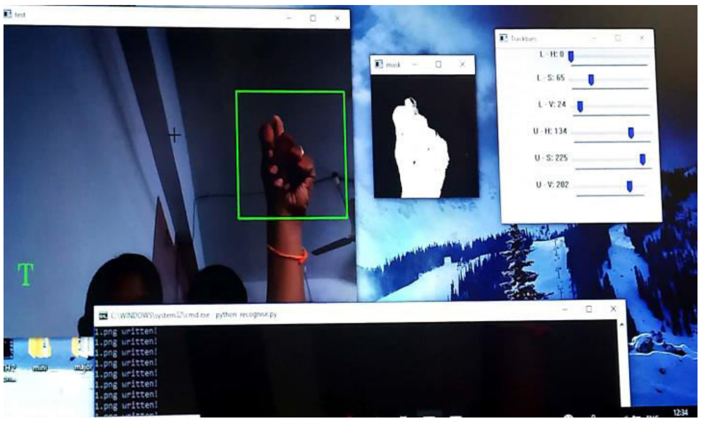
Figure 5.Model Loss

The graphs are plotted for accuracy vs epoch and loss vs epoch. The results showed satisfactory outputs for testing data.

We saw the performances improves differently in our two datasets via data augmentation. By transforming our images just a few pixels there was an increased accuracy. We also flipped the images horizontally. While it wasn't extremely effective, we saw that with better and more representative initial training data, augmenting improved the performance more drastically. This was observed after augmentation, which improve the performance of the model.

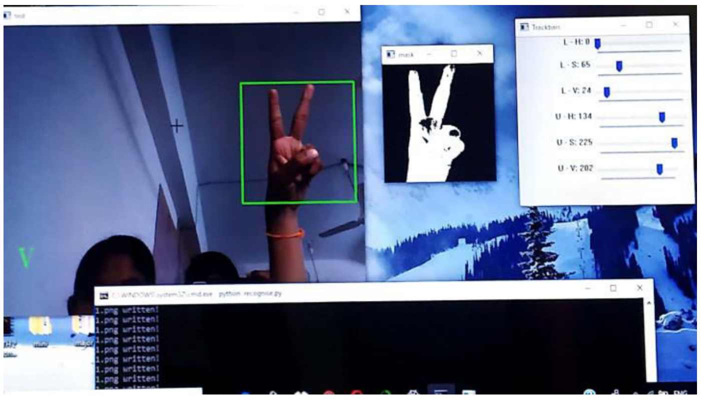
There are three hidden layers and hence with 25 epochs the model is trained rigorously. The learning rate for the SGD is 0.01, the activation function used is ReLU, the optimization technique used is Stochastic gradient descent with learning rate 0.01. The loss function is categorical cross entropy of softmax function.

After training the model the next step is to recognize the human hand. Hence we are using computer's webcam to continuously record the video of input which is taken from a small green square box as shown in the below figure.



```
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
```

Figure 6.Detecting 'T' symbol



```
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
1.png written!
```

Figure 7.Detecting 'V' symbol

As we can see that our hand is detected in the mask CNN window. The mask CNN is environment effected i.e., if in the green box other than hand, if any object occurs the output will be effected. We have put our system disturbance independent by setting the hand only in the box with help of mask CNN background. This output can be improved by changing the HSV values accordingly in the track bar window. The detected letter is displayed in the web cam window.

From figure 6, letter 'T' is detected. The sign code for T is detected in the webcam window and the mask CNN will detect the sign in that green box. The perfect detection will be done by changing the HSV coordinates in the track bar window and similarly an letter 'V' is detected with the help of its sign code. While the computer is continuously capturing the video, in background the model continuously read the values of the green box even though hand is present or not.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, we described a deep learning approach for a classification algorithm of American Sign Language. Our results and process were severely affected and hindered by noise and lighting variations in our self-generated data. This is because we are preparing this model on Mask CNN approach. To overcome this problem we can use RGB model including HSV to improve our results, so that it can detect any real-time images.

In recognizing that classification is a limited goal, we plan on incorporating structured PGNs in future implementations of this classification schema that would describe the probability distributions of the different letters occurrences based on their sequential contexts. We think that by accounting for how the individual letters interact with each other directly, the accuracy of the classification would increase. This HMM approach with sequential pattern boosting has been done with actual gesture units that occur in certain gestures contexts, i.e. capturing the upper arm movements that precede a certain letter to incorporate that probability weight into the next units class, and processing sequential phonological information in tandem with gesture recognition, but not for part-of-word tagging with an application like what we hope to achieve.

We also recognize that the representation itself makes a huge difference in the performance of algorithms like ours, so we hope to find the best representation of our data, and building off our results from this research, incorporate it into a zero-shot learning as having the potential to facilitate the translation process from American Sign Language into English. Implementing one-shot learning for translating the alphabet and numbers from American Sign Language to written English, and comparing it with pure deep learning heuristic could be successful and have the potential to benefit from error correction via language models. Recent implementations of one-shot adaption have also had success in solving real world computer vision tasks, networks using very little domain-specific data, even as limited as single-image datasets. We ultimately aim to create a holistic and comprehensive representation.

REFERENCES

[1] SAKSHI GOYAL, ISHITA SHARMA, S.S. Sign language recognition system for deaf and dumb people. *International Journal of Engineering Research Technology* 2, 4(April 2013).

[2] D. Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol.

294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.

[3] JOYEETA SINGH, K.D. Indian sign language recognition using eigen value weighted Euclidean distance based classification technique. *International Journal of Advanced Computer Science and Applications* 4,2 (2013).

[4] PADMAVATHI. S, SAIPREEETHY.M.S, V. Indian sign language character recognition using neural networks. IJCA Special Issue on Recent Trends in Pattern Recognition and Image Analysis, RTPRIA (2013).

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] K. Elissa, "Title of paper if known," unpublished.

[7] Nmanivas. Gesture recognition system. https://github.com/nmanivas/Gesture-Recogition-System.

[8] UdacityMLND_Capstone/capstone_proposal. http://github.com/UdacityMLND_Capstone/blob/master/proposal/capstone_proposal.pdf

[9] Pugeault, N., and Bowden, R. (2011). Spelling it out:Real-time ASL finger spelling recognition in proceedings of the first IEEE workshop on consumer depth cameras for computer vision, jointly with ICCV`2011.

[10] Pugeault, Nicolas. "Nico" ASL Finger Spelling dataset from the University of Surrey's Center for Vision, speech and Signal Processing, empslocal.ex.ac.uk/people/staff/np331/index.php?section=FingerSpellingDataset.