American Sign Language Classification using Machine Learning Algorithms

Darpan Mehra[1], Ankitha Udupa[2]
Northeastern University
mehra.d@northeastern.edu[1], udupa.a@northeastern.edu[2]

Abstract

Computer vision has several applications such as monitoring, social networking, medical imaging as well as in large scale multi-purpose industries. Along with other multi-purpose applications, computer vision has a widespread utility in providing support for differently-abled people. American Sign Language (ASL) alphabet recognition using computer vision is a challenging task due to the interclass similarities between different feature images, inference occlusion and large intraclass variation. This paper describes a method for American Sign Language Recognition using Support Vector Machines and Convolution Neural Network.

I. Introduction

Learning and interpreting ASL is difficult and there is a dearth in the number of people who can successfully interpret and communicate using sign language. The National Centre for Health Statistics estimates that 28 million Americans (about 10% of the population) have some degree of hearing loss [5].

About 2 million of these 28 million people are classified as deaf which means they can't hear everyday sounds or speech even with a hearing aid. This makes it increasingly important to support the use and therefore complement the existing methods to understand and communicate effectively using the American Sign Language.

As per the research by Garcia & Viesca [1], there are about 250,000 to 500,000 speakers. This is approximately one speaker per fifty-six people requiring support for communication using ASL.

At first, written communication seems to be the best alternative where the speaker and the interpreter use a written mode of communication, however, this can be cumbersome, time-consuming, impersonal and impractical during an emergency situation. To tackle this problem and to provide an improved mechanism to interpret ASL, we propose a solution of American Sign Language recognition using a Convolutional Neural Network. This method is trained on approximately 27455 records of varied alphabets of the ASL.

American Sign Language detection can be broadly classified into real-time classification using computer vision where an input device such as a camera is used to read dynamic fingerspelling detection using static fingerspelling of the American Sign Language which uses the loaded images of various alphabets of the American Sign Language. These images are then used to train and test the Convolution Neural Network model.

In this implementation, we will be using static fingerspelling recognition using Artificial Intelligence. The static gesture recognition poses an incredible difficulty due to visual similarities in different gestures. For instance, the alphabets ‘M’ and ‘N’ almost look identical in the fingerspelling and it is only distinguishable by the usage of the thumb. Our system features a pipeline that takes a sample of an image representing an alphabet of the American Sign Language and outputs a label of the corresponding American Sign Language alphabet.

II. Related Work

Over the last few years, there have been multiple researchers who have used classifiers such as linear classifiers, Bayesian networks and neural networks to classify different alphabets of the American Sign Language. One of the first few approaches of classification was by Bergh in 2011 [2].

Haar wavelets and databases were utilised to build a hand gesture recognition system. This system provided good results however, it only employs the ability to recognize six classes of gestures. Similar classification research has also been carried out on different types of languages such as British Sign Language (BSL).

Previously, Hidden Markov Models were also used to extract features from the images. Hee-Deok Yang, Chosun University, Korea [3] used 3D locations of hands and face to extract depth information using Microsoft’s Kinect. Using this approach for recognizing signs and from signed sentences, Hee-Deok Yang through his research was able to achieve an accuracy of 90.4%.

In this study, we have used Convolutional Neural Networks and Support Vector Machines for American Sign Language classification. We have used data preprocessing techniques like scaling, resizing and grey-scaling images to improve training results. In our experiment, we explored different architectures of neural networks and support vector machine kernels to obtain 95% test accuracy on the American Sign Language MNIST dataset.

III. Dataset

The MNIST (Modified National Institute of Standards and Technology) database is one of the largest and most popular databases for handwritten digits used in machine learning and deep learning. The original MNIST dataset for handwritten digits is extensively used in artificial intelligence.

For our experiment, we have used the sign language MNIST dataset which consists of 27,455 training image data and 7,172 test image data. A sample of the images is shown in Figure 1.

A record of a sample represents the pixel information of a 28 x 28 dimension. This means that the feature is represented by 784 distinct information about each pixel of the sample image. Figure 2 shows that our data is evenly distributed.

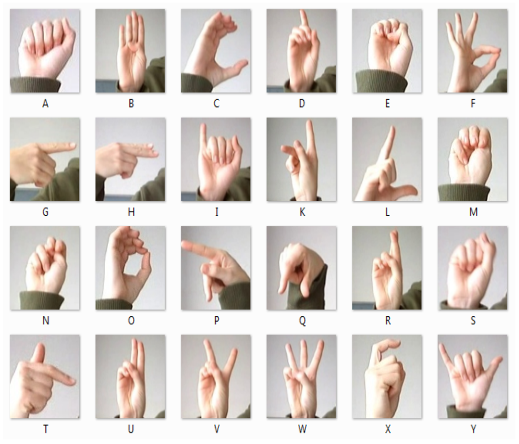

Figure 1. American Sign Language alphabets

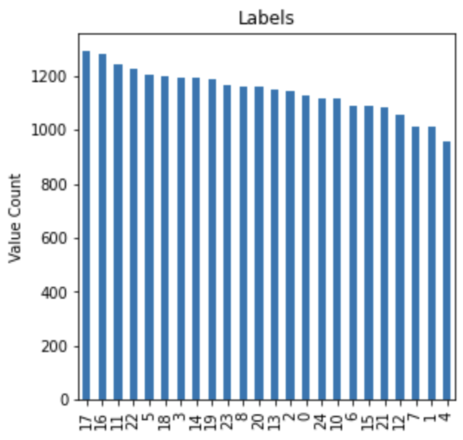

Figure 2. Distribution of data

In this implementation, we will be building a system to classify and recognize 24 alphabets. Alphabets ‘J’ and ‘Z’ are excluded as these alphabets are represented by hand and finger movement.

IV. Background

In our experiments, we have used several different kernels for the Support Vector Machine implementation and different degrees for hidden layers in our Convolution Neural Network implementation.

In Support Vector Machines, kernels are functions that help us solve different kinds of problems. Kernels provide a mechanism to manipulate the data we are using in our implementation. Different types of Support Vector Machines use different types of kernels such as Linear Kernels, Polynomial Kernels, Radial Basis Function Kernels, Sigmoidal Kernels and more.

Linear Kernels are the simplest of them all. It is used when the data is linearly separable i.e. the data can be separated using a single line. Figure 3 illustrates data that is linearly separable.

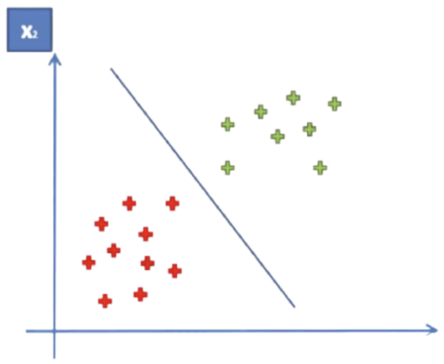

Figure 3. Linearly Separable Kernels

Polynomial Kernels, on the other hand, look not only at the provided feature input set to determine similarities and dissimilarities, but also a combination of these [8]. For degree d polynomials, the polynomial kernel is defined by $K(x,y) = (x^Ty + c)^d$ where x and y are vectors in the input space and c is defined as a free parameter. The value of c determines if the kernel is homogenous or not, which is when $c = 0$.

Radial Basis Function (RBF) kernels are the most popularly used kernels in Support Vector Machines where the function is defined as $\exp(-\gamma||x - y||^2)$. The gamma variable (γ) is a free parameter that decides the amount of influence the given two points have on each other.

In kernels, one of the important factors is the C parameter. The C parameter tells the Support Vector Machine optimizer how much or how less we want to avoid misclassification in each training example. A higher C value will result in choosing a smaller margin hyperplane. The basic idea of the Support Vector Machines is to use as big a margin as possible with as low misclassifications as possible. These at first, may sound like a contradictory statement, but the middle ground helps us solve problems using Support Vector Machines.

We have also used convolution neural networks in our experiments. One of the main benefits of using a Convolution Neural Network is that it automatically identifies the important features without much human intervention. This benefits greatly in terms of time spent preprocessing the data and analysing the important features for feature extraction.

A neural network is based on the idea of how a human brain works. A neural network is a collection of several neurons that takes in an input, processes it, and develops an output. These neurons are organised into different layers and may or may not be computing similar things. A neural network has an input layer at the beginning which is designed to receive different kinds of information from the real world, organise, interpret and classify it. At the end of the neural network lies the output layer awaiting the result of the process. In the middle of these two layers lies a series of hidden layers which is responsible for the most part of the computing and the logic.

The connections between the two neurons are called weights and can be either negative or positive. Neural networks are extremely adaptive and they learn well on their own without much human intervention. Figure 4 illustrates a neural network.

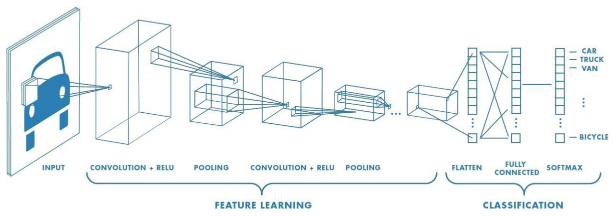

Figure 4: Convolution Neural Network

The output of each neuron in the neural network depends on the activation function used. An activation function is a function that determines the output of the neuron. There are two types of activation functions, namely- Linear activation functions and Non-linear activation functions. In a linear activation function, the output of the function will not be confined between any range. This type of activation function is not widely used in machine learning applications. On the contrary, the non-linear activation functions are the most used activation functions in modern neural networks. This type of activation function allows the model to create a mapping between the network's input and the output. This knowledge is essential to learn and model complex data such as videos, images, audio and data sets with a high degree of dimensionality. The output of each neuron also depends on the type of activation function it is using.

Two of the widely used activation functions are the Sigmoidal activation function and Rectified Linear Unit Activation function.

The Sigmoidal activation function is used as it gives an output in the range 0 and 1. This helps in the models used to determine probabilistic value as it can be closely associated with it. The sigmoidal activation function is differentiable which means we can find the slope given any two points on the curve. The sigmoidal activation function is also monotonic in nature, however, the derivative of the function is not. Figure 5 illustrates the sigmoidal activation function.

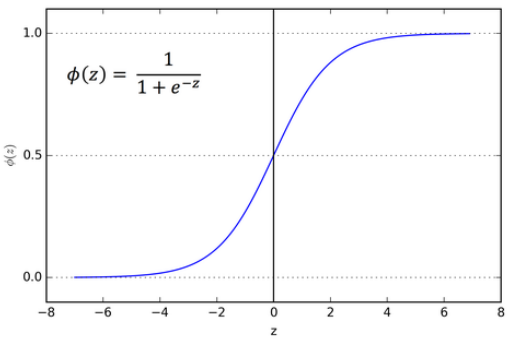

Figure 5: Sigmoidal Activation Function

The Softmax activation function is a generalisation of the logistic function to multiple dimensions that means it can be used in multinomial logistic regression [6]. Softmax is a mathematical function that helps to scale a vector of numbers into a vector of probabilities where the probabilities are proportional to the relative scale of each value in the vector. Such an activation function is most suitable when we have an image with pixel information. Using this we can map it to probabilities. This makes it easier to classify the images in our dataset. Softmax is defined as $S(z)i \;= \frac{exp(zi)}{\Sigma\, exp(zj)}$ where z is an input vector to the function exp(z_i) is the standard exponential function applied on z_i. Figure 6 illustrates an example of the softmax function.

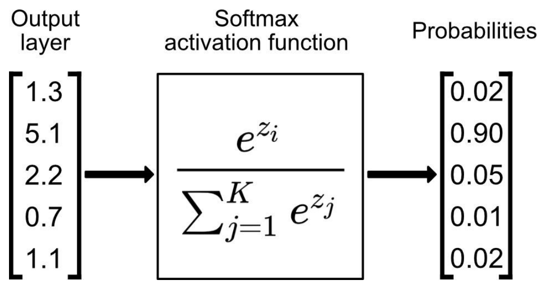
Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

Figure 6: Example of a Softmax activation function

The Rectified Linear Unit (ReLU) activation function is widely used in machine learning applications [7]. ReLU is half rectified from the bottom which means that when *z* is less than zero, *f(z)* is zero and *f(z)* is equal to *z* when the value of *z* is above and equal to zero. The range of the ReLU function is from 0 to infinity. The function and the derivative both are monotonic in nature. One of the issues with the Rectified Linear Unit is that the negative values become zero, which means that any negative input value given to the ReLU activation function turns the value into zero which affects the output graph as the negative values are not mapped appropriately.

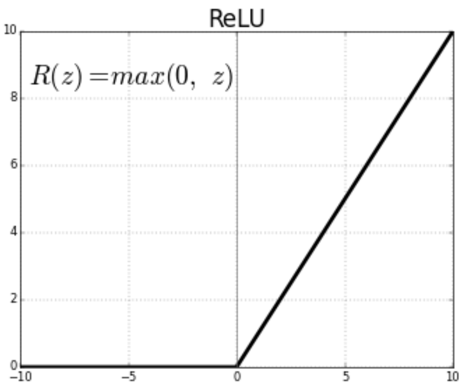

Figure 7: Rectified Linear Unit activation function

V. Methods

In this experiment, we have extracted 27,455 training images and 7172 test images along with their associated labels. The training images are scaled by a factor of 255 and resized to a shape (28, 28, 1) to facilitate better training. We present two methods that result in successfully classifying alphabets in accordance with the alphabets in American Sign Language (ASL). The first method uses a support vector machine and the second method uses a convolutional neural network.

A. Support Vector Machine

Support vector machines are highly preferred for classification tasks as it requires less computational power and produces remarkable results. A support vector machine algorithm finds an optimal hyperplane in an N-dimensional space to classify the data points. We have experimented with multiple different architectures of Support Vector Machine using kernels such as Linear Kernel, Radial Basis Kernel and Polynomial Kernel. We have also explored these three kernels with different regularisation parameters *'C'* such as 0.005, 0.02, 0.08, 0.2, 0.6 and 1 to examine the effects of soft and hard margins for the hyperplane.

B. Convolutional Neural Network

A Convolutional Neural Network is a deep learning technique that is popularly used for tasks such as image classification. A Convolutional Neural Network requires less image preprocessing as compared to support vector machines. The network has the ability to learn the filters and characteristics that otherwise, need to be hand-engineered.

In our study, we have experimented with different Convolutional Neural Network architectures. We have explored the Convolutional Neural Networks with one, two, and three hidden layers along with the input layer and the output layer.

Each layer consists of a 3x3 kernel, a 2x2 max-pool layer and a dropout layer. Initially, the results were achieved by using a dropout with a probability value of 0.2 which led to the model overfitting the data. The dropout probability was then changed to 0.4 for better results. We have also experimented with kernel sizes 3x3 and 5x5 and different numbers of filters for each layer.

The best results were achieved using a two-layer Convolutional Neural Network, two max-pooling layers of size 2x2, three dropout layers with a probability value of 0.4, two dense layers and one fully connected layer.

The first two convolutional layers use Rectified Linear Unit activation (ReLU) function and the final dense layer uses the Softmax activation function for categorical data. The model was trained using a categorical cross-entropy loss with an ADAM optimizer [8].

VI. Experimental Results

Kernel	Accuracy %											
	C=0.005		C=0.02		C=0.08		C=0.2		C=0.6		C=1	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Linear	88.83	71.51	98.04	76.82	100	77.53	100	78.76	100	78.97	100	78.48
Radial Basis	13.39	6.90	57.63	43.23	89.02	70.41	97.63	80.39	99.99	83.85	100	83.82
Polynomial	99.99	79.55	100	78.75	100	77.63	100	78.61	100	78.17	100	78.47

Table 1: SVM Results

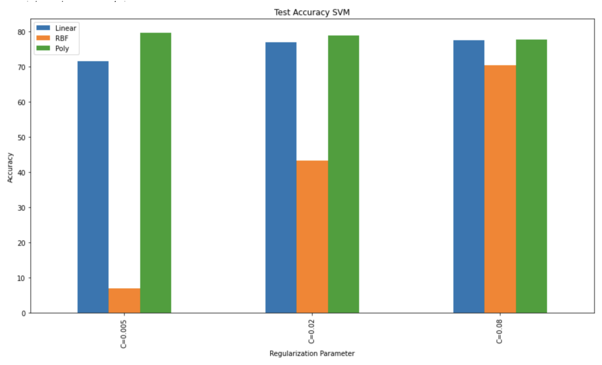

Figure 8. Comparative bar plot of test accuracies for linear, RBF and Polynomial SVM kernels for C < 0.01

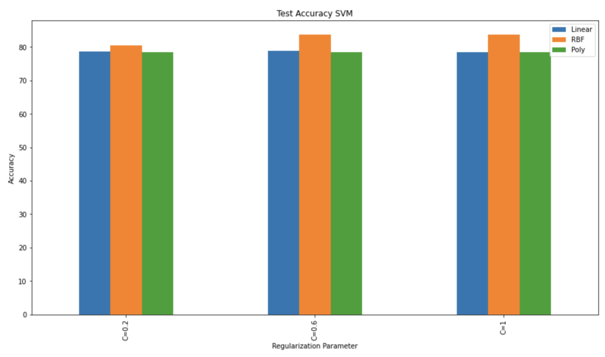

Figure 9. Comparative bar plot of test accuracies for linear, RBF and Polynomial SVM kernels for 0.01<C<1

In this study, twenty-one different architectures of Support Vector Machines were used to achieve sign language classification. Table I represents the result of the different architectures. For *'C'* values 0.005 and 0.002, 0.08 linear and polynomial kernels outperform the radial basis kernel as seen in Figure 8. However, as we increase the *'C'* values to 0.2, 0.6, 1, we see that even though the linear and polynomial kernels have better training accuracy, the test accuracy is quite low as seen in Figure 9. This informs us that these architectures are overfitting the data and not quite suitable. In the case of radial basis kernels, the training accuracy is lesser than the previous architectures but it is not overfitting the data and hence is better suited for our application.

Accuracy %	Architectures			
	Model 1	Model 2	Model 3	Model 4
Training	99.99	100	70.83	99.9
Test	86.41	86.71	59.20	96.23

Table 2: CNN Results

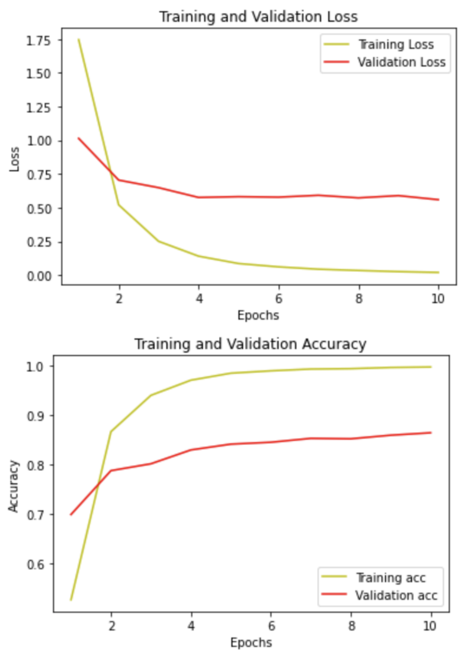

Figure 10. Training and validation loss/accuracy for CNN Architecture 1

Our second approach employs the convolutional neural network to achieve sign language classification. Convolutional Neural Network architectures often require hyperparameter tuning to get optimal results. In our experiment, we have tuned the model with respect to the number of hidden layers, dropout, kernel size and the number of filters. In this study, we have used 4 different architectures to achieve results. Our first architecture for the convolutional neural network approach includes one convolutional layer with a 3x3 kernel, thirty-two filters and Rectified Linear activation (ReLU) function. This layer is followed by a 2x2 max-pool layer and a dropout layer with a probability value of 0.4. This is then followed by a flatten layer and another dropout layer with a value of 0.4 which is then fed to a dense layer with a softmax activation function. As we can see in Table II, this architecture performs well on the training data but does poorly on the validation and test data. This results in our model overfitting as shown by Figure 10.

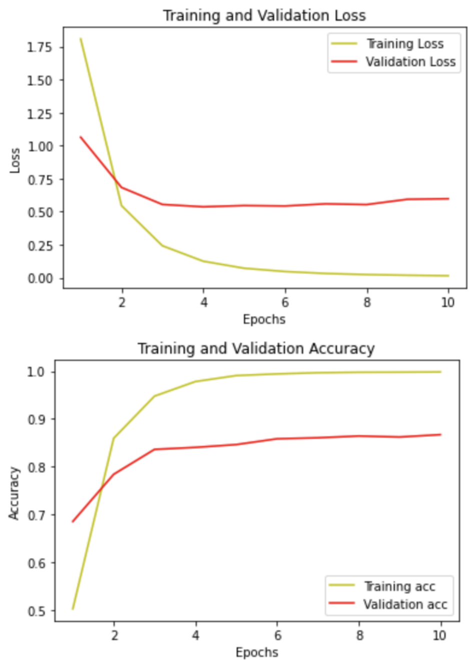

Figure 11. Training and validation loss/accuracy for CNN Architecture 2

In our second architecture, we changed the number of filters in the above architecture from 32 to 64. This does not affect the performance of the model considerably as shown in Table 2. Increasing the number of filters in the input layer does not have much effect on performance as the first layer in the convolutional model detected a small number of low-level features. This architecture still suggests the model is overfitting as shown by Figure 11.

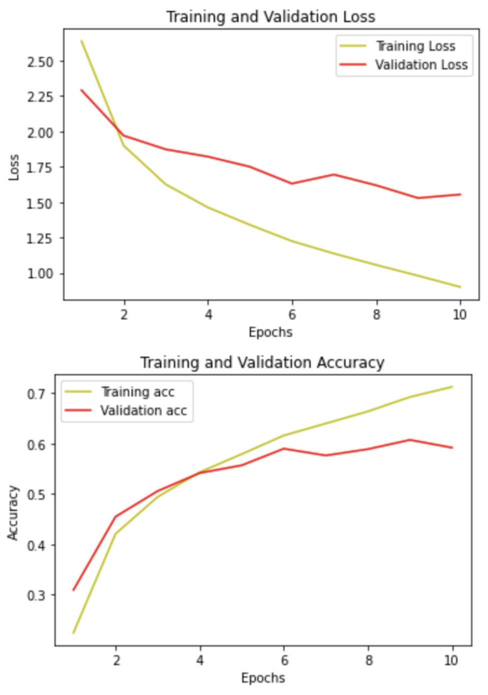

Figure 12. Training and validation loss/accuracy for CNN Architecture 3.

In our third architecture, we added a second convolutional layer with a 3x3 kernel, 64 filters and Rectified Linear Activation (ReLU). This layer is followed by a 2x2 max-pool layer and a dropout layer with probability p=0.2. Table II shows that this model performance degrades on the training data as well the test and validation and test data. A dropout layer with probability p=0.2 degrades our model's performance as the model tends to overfit the training data as we can see from Figure 12.

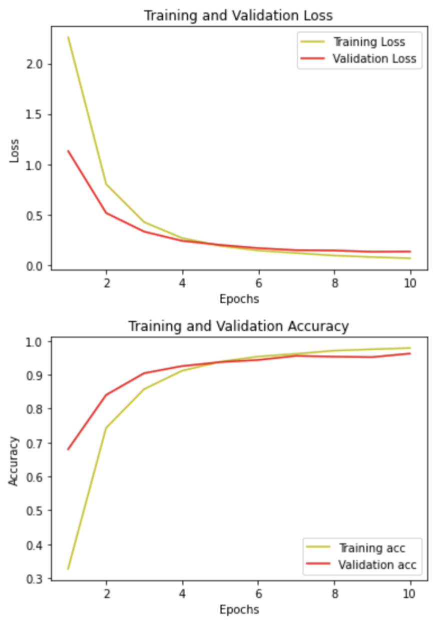

Figure 13. Training and validation loss/accuracy for CNN Architecture 4

In our final architecture, we changed the dropout rate in architecture 4 from probability 0.2 to 0.4. This significantly improved the performance of our model. Table 2 shows that this architecture gives us high accuracy on training, validation and test data without the model overfitting the data as seen in Figure 13. This also informs us that the initial one layer architecture approach could only detect low-level features and was not sufficient to properly classify images. The final results as seen in Figure 14. suggests that the fourth architecture for the convolutional neural network gives us optimal results.

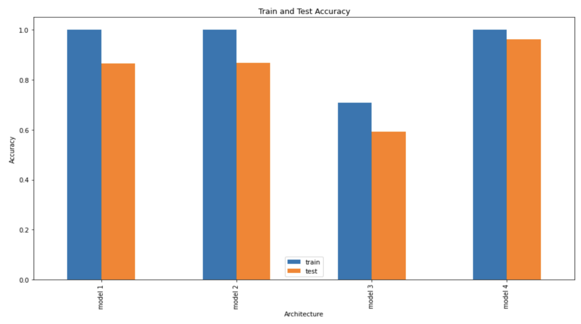

Figure 14. Comparative bar plot of training and test accuracies for different CNN architectures

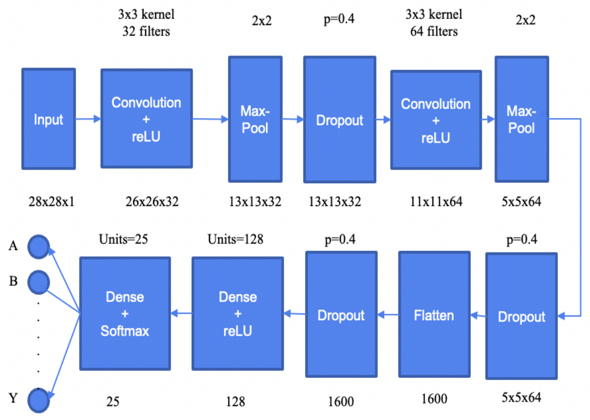

Figure 15. Final CNN architecture

VII. Conclusion

There are an increasing number of people who require support for effective communication. American Sign Language is one of the most widely used sign languages in America by people with hearing and speech imparities.

In our experiment, we conclude that a Convolutional Neural Network with two convolution layers, two max-pool layers, three dropout layers with a probability value of 0.4, one flatten layer and two dense layers as seen in figure 15 give us an accuracy of 95% on test data. This outperforms the support vector machine approach with an average test accuracy of 83.82% for sign language classification. Furthermore, convolutional neural networks require less image preprocessing as compared to support vector machines.

This model builds a foundation that can be further extended to build applications like automatic text generation using sign language hand gestures, real-time sign language recognition and sign language to speech conversion. Such a system drastically reduces the time one takes in learning a language, especially a sign language.

The system can also be further expanded into augmented reality applications where the classification is done in real time. The alphabet recognition and classification can be used as a stepping stone for understanding complete sentences spoken using the American Sign Language. This will enable enhanced real-time communication between a person with a hearing disability and a non-ASL speaker.

We estimate that such a system can be widely used in places such as restaurants, salons, courtrooms, conferences, and more.

References

[1]. Garcia, B., & Viesca, S. A. (2016). Real-time American sign language recognition with convolutional neural networks. Convolutional Neural Networks for Visual Recognition, 2, 225-232.

[2]. Van den Bergh, M., & Van Gool, L. (2011, January). Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In 2011 IEEE workshop on applications of computer vision (WACV) (pp. 66-72).

[3]. Hee-Deok Yang, Department of Computer Engineering, Chosun University, Sign Language Recognition with the Kinect Sensor Based on Conditional Random Fields, Sensors 2015, 15, 135-147; doi:10.3390/s150100135

[4]. T. Starner, J. Weaver and A. Pentland, "Real-time American sign language recognition using desk and wearable computer-based video," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371-1375, Dec. 1998.

[5]. National Institute on Deafness and other Communication Disorders, the year n.d.

[6]. Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, Activation Functions: Comparison of Trends in Practice and Research for Deep Learning

[7]. Abien Fred M. Agarap, Deep Learning using Rectified Linear Units (ReLU)

[8]. Diederik P. Kingma, University of Amsterdam, OpenAI, Jimmy Lei Ba, ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.