

2019

## ASL Reverse Dictionary - ASL Translation Using Deep Learning

Ann Nelson

*Southern Methodist University*, [alnelson@mail.smu.edu](mailto:alnelson@mail.smu.edu)

KJ Price

*Southern Methodist University*, [kjprice@mail.smu.edu](mailto:kjprice@mail.smu.edu)

Rosalie Multari

*Sandia National Laboratory*, [ramulta@sandia.gov](mailto:ramulta@sandia.gov)

Follow this and additional works at: <https://scholar.smu.edu/datasciencereview>

Part of the [Applied Statistics Commons](#)

---

### Recommended Citation

Nelson, Ann; Price, KJ; and Multari, Rosalie (2019) "ASL Reverse Dictionary - ASL Translation Using Deep Learning," *SMU Data Science Review*: Vol. 2 : No. 1 , Article 21.

Available at: <https://scholar.smu.edu/datasciencereview/vol2/iss1/21>

This Article is brought to you for free and open access by SMU Scholar. It has been accepted for inclusion in SMU Data Science Review by an authorized administrator of SMU Scholar. For more information, please visit <http://digitalrepository.smu.edu>.

# ASL Reverse Dictionary

## ASL Translation Using Deep Learning

K.J. Price<sup>1</sup>, Ann Nelson<sup>1</sup>, Rosalie Multari<sup>2</sup>

<sup>1</sup> Master of Science in Data Science, Southern Methodist University,  
Dallas, TX 75275 USA

<sup>2</sup> Sandia National Laboratories, Albuquerque, NM 87123

**Abstract.** The challenges of learning a new language can be reduced with real-time feedback on pronunciation and language usage. Today there are readily available technologies which provide such feedback on spoken languages, by translating the voice of the learner into written text. For someone seeking to learn American Sign Language (ASL), there is however no such feedback application available. A learner of American Sign Language might reference websites or books to obtain an image of a hand sign for a word. This process is like looking up a word in a dictionary, and if the person wanted to know if they were doing the sign correctly; or to know what a sign means, there is no way of checking. Because of this, the automated translation of ASL has been an active area of research since the early 2000's. Researchers have investigated numerous ways of capturing hand signs, as well as numerous methods for recognizing or categorizing the captured data. In this work, we demonstrate an ASL translation system based on Convolutional Neural Networks (CNN), that provides a practical application for the real-time translation of 29 ASL signs, including 26 alphabet signs and three additional signs ('space', 'delete', and 'nothing'). This application can be used to translate a hand sign for a person learning ASL as well as to facilitate communication between an ASL-signer and a non-signer. The CNN model used in this study is based on the Keras VGG16 [1] pre-trained model and pre-processed images. It has 100% accuracy when predicting on a hold-out/cross-validation testing dataset. The keys to achieving this high precision in automated sign translation are 1) good input images, 2) starting from a pre-trained model 3) fine-tuning of the model. This paper discusses the use of contrast limiting adaptive histogram equalization (CLAHE) image pre-processing to enhance the input images, provides a high-level overview of convolution neural networks (CNN), discusses the use of the VGG16 [1] pre-training model as a starting point for the CNN network and the fine-tuning of the resultant model, and provides an overview of the web application implemented for real-time ASL translation. The results of experiments used to assess the strength and generalization capabilities of the model are also detailed.

**Keywords:** Language Translation, Reverse Dictionary.

## 1 Introduction

The World Health Organization estimates that there are about 466 million deaf people in the world (about 5% of the world's population), with 34 million of them being children (March 2018) [2]. They further estimate that, by 2050, over 900 million people will have a disabling hearing loss.<sup>3</sup> There is estimated to be about 500,000 deaf or hard of hearing people in the United States today who use American Sign Language (ASL) as their primary language for communication [3]. ASL is also used by many people who suffer from aphasia (the inability to speak) due to a stroke or other brain damage. The number of aphasic ASL users is not known. Most hearing people do not know ASL. Much work has been done to facilitate communication between the individuals who are deaf and those who are not. The current interpretative technologies available are non-automated, third-party people-centric (in-person or via remote video) solutions. Many people seek to learn ASL, so they can communicate with their ASL-signing co-workers, or family members. Many schools and community programs offer ASL-language classes. However, the independent study of ASL is hampered by the lack of feedback technology for accessing the learner's proficiency and accuracy of signing.

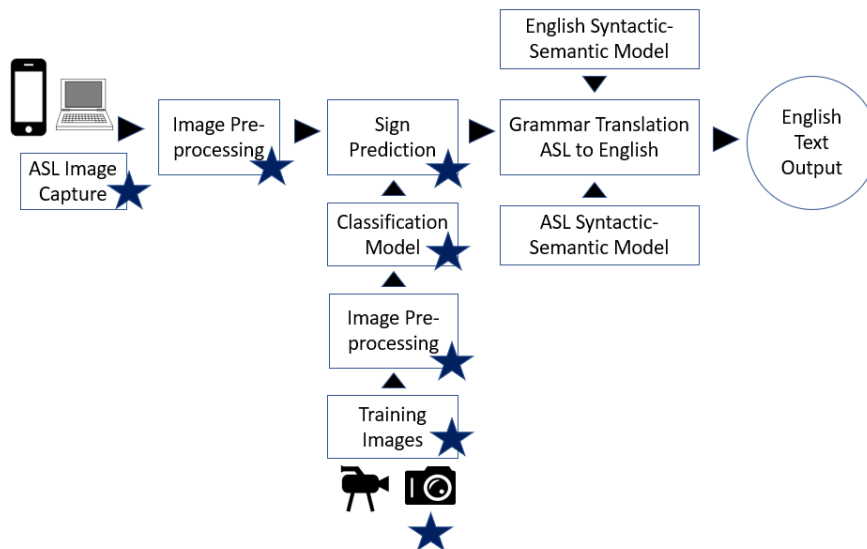
A significant amount of research has been done to develop automated methods for translating ASL from sign to text. Using Deep Learning, we demonstrate a sign language translation system that provides high accuracy in the simplest way possible, while increasing the vocabulary in a scalable way using a Convolution Neural Network (CNN).

ASL was first introduced in 1814, by Dr. Thomas Hopkins Gallaudet [4]. This visual, gestural language is the preferred communication method of the Deaf community and has its own vocabulary and grammar, which is different from English. It was developed by Deaf people to communicate with Deaf people. There are three primary forms of sign language in use in the United States: ASL, Pidgin Signed English (PSE), and Signed Exact English (SEE). PSE was developed to bridge the communication gap between hearing and non-hearing individuals. The hand gestures used in PSE are taken from ASL, but the sentence structure is in English word order. Often a PSE user will speak or mouth words while signing. When sign language translation is provided in a public forum, PSE is typically being used, as the translator will be saying or mouthing English words and grammar while signing. SEE was developed in the 1970s, to help deaf children learn English. It is a complete representation of the English language, and therefore the easiest for parents of deaf children to learn. SEE is usually accompanied by verbalization of the words. Many of the SEE signs are based on ASL with prefixes, suffixes, tenses, and adding "-ing" to words. Because of the structural differences, ASL is not accompanied by spoken English and can be difficult for English speakers to learn.

Co-workers, friends and family members of people who are deaf often seek to learn ASL in order to aid their communication. There is no technology available today which will help the student understand if they are performing ASL signs correctly. Currently, there is no home-use application for translating ASL. However, a business-level system was introduced on Q1 2017, which provides sign-to-voice translation and voice-to-text providing real-time secure and private communication [5].

At the beginning of the project, we envisioned building a web-based system that would provide real-time sign-to-text classification along with ASL-to-English

translation (dealing with the differences in grammar between the two languages). Given the scope of the work needed to develop the fully envisioned system and the given time constraints for this study, we have limited the scope of the work to a subset of the fully-envisioned system, diagrammed in Figure 1.



**Fig. 1.** Schematic of the fully-envisioned image processing system for sign language translation. The components denoted by a star (★) have been incorporated in this study.

For the fully-envisioned system, we believed a CNN-based classification model would be highly effective for the image classification portion module. Natural Language Processing (NLP) and Natural Language Understanding (NLU) would be used to improve sign identification and re-orient grammar and sentence structure to clearly translate ASL to English. The most straightforward manner is to translate SEE (as the grammar matches that of English) however, SEE is not the primary language used by the Deaf. Because of the larger user-base, as well the NLP challenge with grammar differences between ASL and English, we decided to pursue translating ASL to English.

The translation system we developed uses deep learning techniques to identify the 26 finger-spelling alphabet signs and three additional signs ('space', 'delete', and 'nothing'). In prior research automated translation has been accomplished using algorithms built with methods such as Support-Vector Machine (SVM)<sup>6</sup>, K-Nearest Neighbors (KNN)<sup>7</sup>, and Haar-like<sup>8</sup> methods, to name a few. Only one previous study used CNN. Koller and Bowden<sup>9</sup> used paired captured video images with closed-captioned text, to loosely label the video images. They were able to achieve about 50% test accuracy with a CNN model.

This work focuses only on the first stage of the fully-envisioned system, translation of ASL signs, so the next steps to accomplishing the fully-envisioned system would be to employ Natural Language Processing (NLP) and Natural Language Understanding

(NLU) to extend the translation and understanding of the words, into phrases and contextual meaning, thereby enabling the restructuring of the ASL grammar to match English grammar. After this, the next stages would be to 1) add hand position detection within the ‘active space’ of ASL signs, 2) enable the translation of motion-based signs and 3) utilize natural language processing and natural language understanding to interpret ASL grammatical structure.

In this work, we accomplished the translation of hand signs by 1) using the KAGGLE Hand Signs database [10] to train the application to identify finger spelling. The CNN model provides 100% accuracy when predicting on a hold-out/cross-validation testing dataset for the classification of 29 ASL signs, including 26 alphabet signs, and three additional signs (‘space’, ‘delete’, and ‘nothing’).

The JavaScript-based web application ( <https://asl-dictionary.net/> ) developed in this project, captures the hand position of the user. It has a frame capture rate of 1 frame per second. The application uses the video camera on the user’s system, making the application easily accessible, easy-to-use, and economical. After capturing the image, the data is passed through the CNN model, and the top three predicted values of the imaged hand sign and their confidences values are reported. As the user shifts their hand position, the reported predictions shift, encouraging the user to find a hand sign position which achieves a high confidence for the correct prediction. The web application is sensitive to lighting and background noise. Performing the signs in an environment with uniform general lighting or backlighting, and with a neutral background, produces the best results. The accuracy of the application proved difficult to determine and was only qualitatively assessed.

## 2 The Project Work

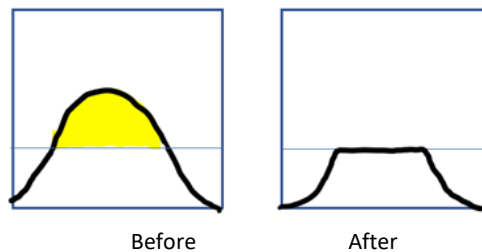
### 2.1 Background

#### 2.1.1 Image Pre-Processing

An automated image pre-processing method was used to programmatically adjust the contrast of the original images of our hand sign dataset. Histogram equalization (HE) [11], normalizes the brightness within images. With HE, areas in an image area with low local contrast gain a higher contrast, while areas of high contrast are diminished. HE works particularly well to enhance details in images that are over- or under-exposed. HE is not resource-intensive, and the original image can be recovered if the equalization function is known. A disadvantage of HE is that this method does not distinguish between noise and the desired features. For our study, we used a variant of HE, contrast limiting histogram equalization (CLAHE) [12]. This method has the advantage of improving local contrast without shifting the average brightness of the image. CLAHE also enhances the definition of edges without overamplifying noise in areas with low contrast, thereby preventing loss of key details. This is accomplished by performing equalizations on local, sub-areas of the original image, placing a higher priority on the local area histograms than on the full image histogram.

The CLAHE algorithm transforms each pixel within an image using a transformation function that is influenced by its neighboring pixel region. The transformation function is proportional to the cumulative distribution function (CDF) of the pixel values in the pixel neighborhood. The resulting value of the pixel is proportional to the slope of the neighborhood CDF at that point. Overamplification can occur in ordinary adaptive histogram equalization (AHE) [12] in pixel neighborhoods with little variation in contrast since the histogram in these regions would be narrow and tall. As a result, AHE may cause noise to be overamplified in these areas. CLAHE prevents overamplification by limiting the histogram to predefined value before computing the CDF, thus limiting the slope of the function. The predefined slope-limiting-value depends on the size of the pixel neighborhood.


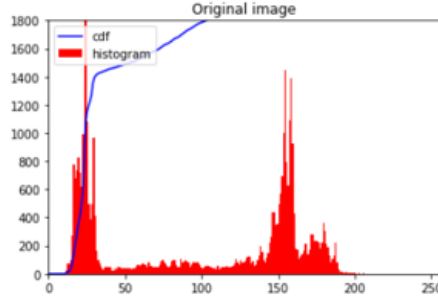

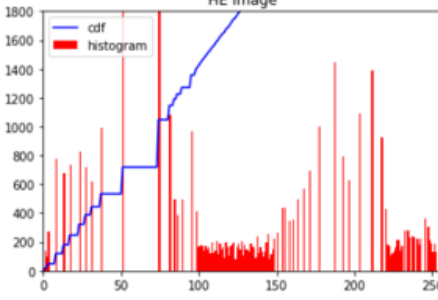

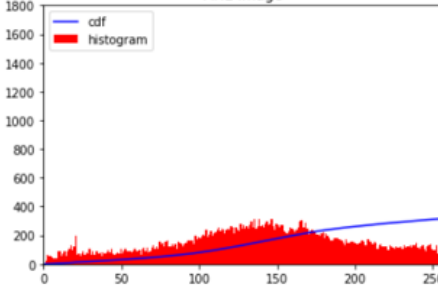
Pixel Brightness Distributions Illustrations  
Before and After CLAHE Implementation

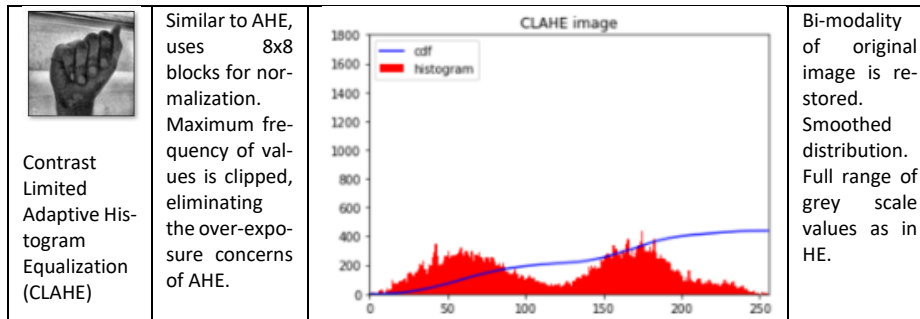


**Fig. 2.** Pixel brightness distributions before and after the application of CLAHE demonstrating contrast normalization and clipping of contrast amplification.

To understand the impact of histogram equalizations, various algorithms were studied to determine the best method for image pre-processing of the dataset. HE on grey scale images is discussed below for image visualization and ease of understanding, however HE on both grey scale and full-color images were used for this study.

**Table 1.** Comparisons of histogram equalization algorithms<sup>13</sup> to demonstrate the advantage of CLAHE for normalizing the contrast across an image, without creating over-exposure in areas of high contrast.

Image / Algorithm Name	Histogram equalization algorithm	Brightness histogram	Observations/ Discussion
 Original Image	None		Bi-modal peaks are the hand (left) and the background (right).
 Histogram Equalization (HE)	The brightness of the image is normalized using the full image.		Bi-modality of the original image remains. Range of values is expanded to the full grey scale range. Gaps in grey scale values created.
 Adaptive Histogram Equalization (AHE)	The brightness values are normalized using 8x8 blocks, across the image. In blocks where there are white value and dark values, the normalized white values are over exposed.		Bi-modality of the original image has been removed. Full range of grey scale values as in HE. Overall distribution is smoother.



### 2.1.2 Convolutional Neural Network Overview

A neural network [14] is a statistical modeling algorithm that can be visualized as a structure of nodes and interconnections between the nodes. The nodes are typically organized into layers. Each layer is assigned an activation function. For neural network modeling, a pattern of inputs is presented to a layer, the operation of the activation is performed on the pattern, by the layer, and a new pattern is output. The output pattern of the layer is then passed to the next layer via the interconnections. The interconnections are defined by learning rules which modify the weights of the connections, and the process is repeated. The weights of the nodes are adjusted programmatically based on the learning rules and the patterns presented to them from the layers. The adjustment of weights is the actual operation of the neural network. Through these structures (interconnects and layers), and mechanisms (learning rules, adjustable weights, and activation functions), the neural network can iteratively, through multiple cycles (or epochs) tune itself and learn to recognize patterns. A neural network may require many epochs to obtain an optimized solution.

All neural networks contain an input layer, and output layer and at least operational layer or hidden layer. A neural network with more than two or three hidden layers is considered a deep learning algorithm. Convolutional neural networks (CNN) have more than 2 hidden layers. The design of a CNN makes it typically well-suited for two-dimensional inputs, therefore image classification is a common application of a CNN as is natural language processing. The typical activation functions (for the hidden layers) in a CNN are pooling, convolutional and fully connected (or densely connected).

Pooling layers combine outputs of multiple nodes in the prior layer into a single node in the next layer. There are two pooling options, max pooling and average pooling. Max pooling passes the maximum value of the set of input nodes and average pooling passes the average value from the set of nodes to the output.

Convolutional layers apply a mathematical operation (the activation function) to two functions to the input. The mathematical operation is specified via the definition of filters assigned to the layer. The functions are created by the input matrix and the filter matrix, the operation is typically a dot product between the two matrices. In a convolutional layer, each node receives input from a subset of input nodes. The subset nodes are defined by the shape of the input area.

Fully connected (dense) layers perform mapping between input nodes and output nodes. In a dense mapping, each input node(n) is mapped to each output node(m)



resulting in  $n * m$  connections to each node in a fully connected layer. Sometimes the activation functions of these layers are adjusted with a biasing term (a real number). The bias term is incrementally adjusted during the learning/training mode process.

Keras [15] is a Python library targeted for deep learning. The Keras API runs on top of TensorFlow [16], Theano [17], and CNTK [18]. Keras [19] facilitates the development of deep learning models by providing powerful functions and statements which have a simple syntax. It also runs on CPU or GPU which can speed up execution. TensorFlow [20] is an open-source library originally developed by Google which enables high performance, numerical computing. It is targeted for deep learning and machine learning. Theano was invented by the University of Toronto and might be considered as the precursor to TensorFlow<sup>21</sup>. For this project, we used Keras and TensorFlow for deep learning modeling.

A typical processing flow for CNN modeling in Keras is reviewed in Table 2.

**Table 2.** A typical process flow for CNN development summarizing the development process steps, their purpose and key objectives.

Process Step	Notes
Input data	Data is typically images or .csv files for image processing and articles or .csv files for NLP.
Create training, validation and test data sets	The validation data set is a subset of the training data set. The validation set is used to demonstrate how well the trained model can generalize against new data the model has not seen yet.
Apply pre-processing	Adjustments in image sizes and image characteristics such as brightness can be applied.
Image normalization	Input image values span 0 (black) to 255 (white). To prevent the lighter images from having extreme influence on the analysis, the values are rescaled to fall between 0 and 1.
CNN model definition	Definition of each layer in the model: the number of files, connections, weights, biases and activation functions. Can be an iterative process to ensure optimized data usage.
Model compilation	Initializes the model for training. This is where the optimizer function and metrics would be declared.
Model training and validation	Defines the number of epochs, and model performance measures of interest. Can take a significant amount of time. GPUs are often used to reduce the time requirement.
Model testing	Confirmation of the effectiveness of the model on observations that were not used for training. The key model performance measure of usefulness. Can drive model adjustments.
Model evaluation	Check for over-fitting and under-fitting. Can drive model adjustments.

CNN modeling research can be quite time-consuming because of the number computations per epoch, the number of epochs per model run, and the number of model iterations to obtain optimal model performance. Using a GPU rather than a CPU to execute the analysis, can reduce the time spent waiting for computations to complete. There are two primary options for enabling GPU usage. In the first, a high-powered, gaming computer, can be used with an on-board GPU. If a high-powered GPU is not locally available, a cloud-based option such as Amazon's AWS EC2<sup>22</sup>, which offers a choice of multiple GPU configurations, or Google's Colaboratory<sup>23</sup> could be used. Python can be run via a locally-based Jupyter notebook configured to access the EC2 cloud space. Colaboratory is free and is accessed via a cloud-based Jupyter notebook and the GPU is configured via a menu dropdown within the notebook. For this project, much of the CNN was run either locally using CPU or GPU or run on Colaboratory. Colaboratory was selected because of its pricing and ease-of-use.

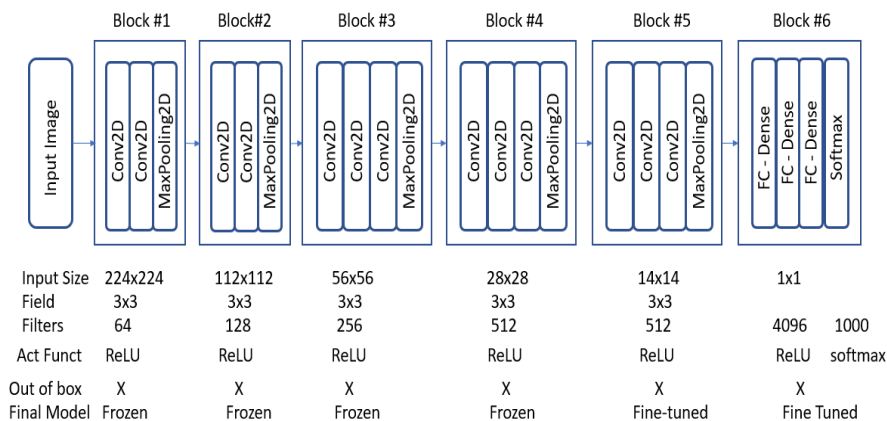
### 2.1.3 VGG16 CNN Model

The VGG16 [24] model was the first and second place winner of the 2014 ImageNet Challenge for image classification. This model has been proven to generalize well to other image datasets. This model incorporates several features that are keys to ensure high accuracy including: 1) utilizing a small receptive window size and a small stride (1 pixel) in the first convolution layer [22]. 2) increased depth [20], which is possible by using a 3x3 filter window in all of layers.

VGG16 is one of five pre-trained CNN models included with the Keras module. VGG16 consists of 6 blocks, each having a 3x3 receptive field, and ReLU as the activation function (see Figure 3). Blocks 1 & 2 have a pair of Convolution2D layers followed by a MaxPooling2D layer. Blocks 3, 4 & 5 have an additional Convolution2D layer. The 6<sup>th</sup> block is a classifier with three fully connected dense layers followed by a softmax layer. The weighted layers are the 13 Convolution2D layers, and the 3 Dense Layers, the sum of which are designed in the VGG16 name.

VGG16 was instrumental in demonstrating that CNN classification and localization accuracies could be improved by increasing the depth of the network. Prior to VGG16, neural networks were typically designed with larger receptive fields and were much shallower than VGG16. VGG selected 3x3 as the receptive field size, because it was the smallest size which enables directional information (up/down, left/right & center). By stacking 3 Convolutional layers in Blocks 3,4 & 5 developers were able to obtain the same effective receptive field as a 7x7 convolution layer. The selection of the non-linear ReLU as the activation function provides protection against vanishing gradients, a problem that can be seen when using a sigmoid function.

After an initial training pass of using the pre-trained VGG16 model, a final model tuned specifically for the training data set can be obtained by freezing the early layers and re-training the model with adjustments in block 5 & 6. This process, called fine-tuning, results in a specific CNN model for specific situation.



**Fig. 3.** Block diagram of VGG16 CNN<sup>25</sup> model composed of 6 blocks, 5 of which are comprised of 2 or 3 Convolution2D Layers with 3x3 receptive fields for feature identification, a Max Pooling 2D layer for parameter reduction, and ReLU as the activation function. The number of filters for each block-layer are labeled and correspond to increased depth. The 6<sup>th</sup> Block is the classification layer. The blocks that were frozen or tuned to complete the model are also labeled.

### 2.1.4 Prior ASL Modeling Work

Multiple ways of collecting the data related to the action of signing have been reported by researchers. Input devices which have been explored include video cameras [26], gloves [27], and Surface Electromyographic (sEMG) signal generators [28]. In 1993, Fels et al. [27] were the first to use a glove device (Cyber Glove) to generate signed words. They were the first group to work specifically on a sign language recognition system and were able to classify 203 gestures with an accuracy of > 99%. In 1995, Liang et al. [29], extended sign language translation research by using the Cyber Glove to recognize alphabet signs via a rule-based voting algorithm. In 2002, Mehdi et al. [30], used the Cyber Glove and a neural network classifier to discriminate 24 ASL characters and 2 special characters. ('J' and 'Z' were omitted because these letters require wrist movement which was out of scope for their analysis process), it should be noted that the performance of their system was poor, as there was no pre-processing of the data prior to feeding the data into their classifier.

Research on image processing has a long history. The seminal research for automated object identification was the 2001 work of Viola and Jones [31] who developed an algorithm which detected human faces in real-time. This work is cited by many papers and is considered the starting point of image processing. Image processing for sign language recognition started in 1998, when Starner et al. [32] mounted a camera to a cap in order to capture hand tracking. They concluded that a cap-mounted camera was better than a desk mounted camera based on accuracy. From 2009 – 2014, several researchers explored using the Kinect camera, used by Microsoft in their Xbox gaming system, for sign language recognition. The Kinect camera was selected because it provides depth data for 3-D imaging thereby reducing the noise and interference of lighting

differences and image backgrounds. Otiniano-Rodriguez et al. [33] focused on comparing Kinect RGB, Depth, and RGB-Depth performances. They found that combining RGB and Depth information gave better results than using either individually. Chuan et al. [34] used a Leap Motion Sensor as their imaging device. This device was selected for its economy, ease of use, and scale. In 2014, Raut et al. [35] used the dataset of 312 ASL hand gestures images. Their results were poor due to image quality associated with lighting issues.

Celal Savur [28] studied the use of sEMG to capture ASL signs. sEMG uses differential voltages associated with muscle movement. It was initially used to control a robotic arm. sEMG signals are collected by using sensors attached to the skin. Theoretically sEMG could perform better than a camera-based by eliminating imaging issues such as noisy backgrounds and lighting, which can make the use of image sources difficult.

Several analysis techniques have been employed to translate sign language. In 1993, Fels [27] used a neural network classification method for ASL (99%+ accuracy). Extreme Learning Machine (ELC) algorithm was selected by Sole et al. [36] for Australian Sign Language classification (95% accuracy). ELC was selected as it is a simplified version of a neural network (NN). Starner et al. [32] used Hidden Markov Models (HMM) to recognize sentence-level ASL. Keskin et al. [37] also used a Random Forest Decision Tree (RF) and achieved 99.9% using Kinect camera inputs. Chuan et al. [8] used k-nearest neighbors (KNN) and SVM to classify 26 letters of American Sign Language. 2014, Raut et al. used LBG Vector Quantization [38]. Fast Fourier Transforms (FFT). PCA, wavelet transforms with PCA have been used for classification of sEMG signals as well.

Because of the ubiquitous availability of video cameras today, we have elected to use webcams or cell phone cameras in our fully envisioned sign language translation system. Based on the learnings of prior researchers, we will be using available databases as input and must be very cautious of the image quality. We will be using CNN models for image classification due to its demonstrated performance in image classification work. This problem was addressed by using several pre-processing techniques to enhance the original images.



## 2.2 Ethical Challenges

The images provided in the Kaggle dataset is strictly from one male, Caucasian subject. Models created using these images have performance bias toward Caucasians because the model may not be robust enough to deal with genetic variations. To alleviate this concern, Keras image augmentation as well as training on pre-processed images using grey scale and contrast limiting adaptive histogram equalization (CLAHE) was used. The effectiveness of addressing left-/right- handed biases using image augmentation was assessed by inverting the test set images (on the x-axis) and then evaluating the prediction accuracy of the model with and without the image generator augmentation.

The effectiveness of addressing additional genetic biases, such as skin tone, hand shape, etc., through the use image augmentation was not able to be assessed due to the lack of availability of test images.

While working on this project a concern was expressed by a Deaf person that we should not be trying to ‘fix’ Deaf people by developing a translation tool. This purpose of the fully-envisioned system is to provide an aid for learning ASL. There is a secondary usage as a communication tool, facilitating communication between users of two different languages. There is no implication that there is a problem with sign language or users of sign language. To alleviate concern that there will be a perception that the application is meant as a “fix” for Deaf people, care must be taken in the wording of marketing communication and any document associated with this application.

**Table 3.** Experimental results assessing the right-/ left- hand bias and the ability of Keras augmentation to address the bias concern. Use of Keras’ image generator, horizontal flipping option, effectively eliminates concerns of right-handed bias in the model.

	image	Right-hand Model built without horizontal flipping in image generator (test accuracy)	Right hand Model built with horizontal flipping in image generator	Right- & Left-hand Model built without flipping	Right- & Left-hand Model built with horizontal flipping
Right-handed test images		.991	.994	.966	.985
Left-handed test images		.473	.987	.960	.992

As in any interpretation or translation work, accuracy of the translation is paramount in order to correctly express the intent of the signer. For a community that has experienced disenfranchisement in the hearing community, incorrect translation of their intended communication is a serious concern and could be perceived as another injustice. To alleviate this concern, the application must achieve high translation accuracy.

### 3 Methods

Important to the success of this work, was the addition of image pre-processing to improve the image quality of the incoming images. The Kaggle hands data set, for instance, provides about 3,000 images for each hand sign. The images vary in the lighting conditions as well as the hand placement within the image field. The lighting of the image has a large impact on the ability of the computer to identify features within the image, which is crucial for classification accuracy. Examples of the original images obtained from the hand sign dataset for the letter ‘A’, are shown in Figure 4.



**Fig. 4.** Examples of images of the letter ‘A’ within the Kaggle hand sign data set demonstrating a variety of lighting conditions.

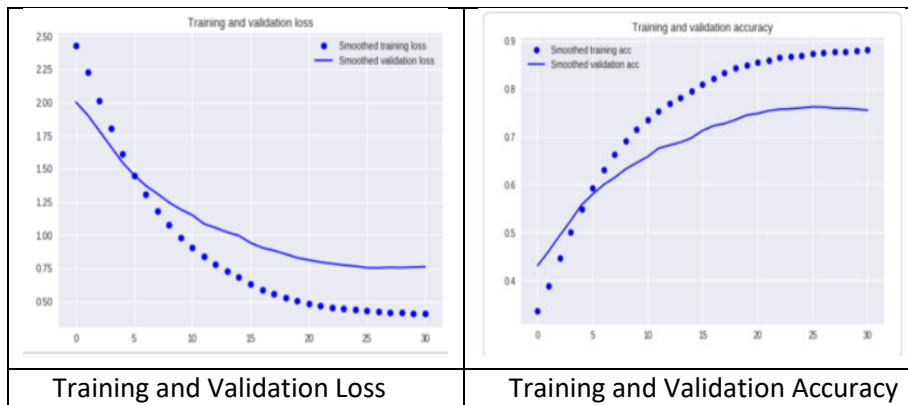
In order to ‘clean’ the data and improve the image recognition and classification accuracy of the system, five additional version of each of the original images were evaluated using a different image pre-processing algorithm provided by OpenCV<sup>39</sup> (an open source computer vision library). The CNN model used a random sampling of 5 versions of each image(4A-4E). Table 4 shows six of the pre-processed image variations explored.

**Table 4.** Examples of the pre-processing methods evaluated to address image quality. GS= Grey Scale, CLAHE = contrast limiting adaptive histogram equalization, NEG = negative. The order of the processing naming is the order of the application, that is GS-CLAHE means a grey-scale image of the original is generated and then CLAHE is applied to the grey-scale image. CLAHE is a key pre-processing step to enable high precision. The order of the processing doesn’t matter (comparing 4E & 4F).

Image						
Image #	4A	4B	4C	4D	4E	4F
Pre-Processing Description	Original	GS-CLAHE	FC-CLAHE	GS-CLAHE-NEG	FC-CLAHE-NEG	FC-NEG-CLAHE
Test Accuracy	98.96%	99.43%	100%	99.6%	100%	100.00%
Run 1-3	98.9%	98.86%	99.43%	98,29%	99.43%	99.43%
	96.6%		100%			

Table 4 Image # 4A is the original image provided by the Kaggle hand sign dataset. Image 4B is a contrast limiting adaptive histogram equalization (CLAHE) of a grey scale (GS) image of Image 4A. Image 4C is the CLAHE version of the full-color (FC) original image 4A. Image 4D is the negative (NEG) of Image 4B and Image 4E is the negative of Image 4C. Image 4F, though created by reversing the order of the processing steps used to create Image 4E, ends up identical to Image 4E. All images are 200x200 pixels in size. In addition to this image manipulation pre-processing, image augmentation of the images was applied to the training and test datasets, via the Keras image\_generator() function. The combination of these methods resulted in a training dataset of 317,000 images from which a random sample was selected and used for training the model. The test data was generated by random sampling 10% of the original images, prior to the pre-processing. The test dataset received the same pre-processing and augmentations as the training dataset. The validation data set was a random sample

of 10% of the training images. The test accuracy for each of the pre-processing sets is listed below each image. Over-fitting was prevented by using conservative callbacks on the validation loss rate. Fig. 5 shows the training and validation test losses and accuracies demonstrating that overfitting has been minimized. Note that the training stopped after 30 epochs when validation loss and validation accuracy curves reached their optimums.



**Fig. 5.** Training and validation test losses and accuracies demonstrating that overfitting has been minimized. Note that the training stopped after 30 epochs when validation loss and validation accuracy curves reached their optimums.

Model development began by using the Keras VGG16 with weights pre-trained on ImageNet model. Added to the VGG16 model, were 3 custom final dense layers. The initial training on the 5 input images provided 98.96% accuracy, with the best test accuracy using CLAHE pre-processing. The initial input layers of the model were frozen, layers 17-19 were tweaked and the model retrained. This model provided 100% test accuracy.

**Table 5.** Evaluation of the model pre- and post- fine tuning against original images and the original + pre-processed images.

Data Set	Test Accuracy – VGG16 without fine tuning	Test Accuracy - VGG16 model with fine tuning
Original images	97%	98%
Pre-processed images	92%	100%

### 3.1 Experiments

An experiment evaluating the performance of a blank models, to VGG16 and VGG19 was performed. VGG16 was found to outperform VGG19 by about 5%. Both of the pre-trained models were a vast improvement over our own blank models.

**Table 6.** Comparison of VGG pre-trained models to blank models using original images. The VGG16 pre-trained model was selected as the foundation for the CNN model.

Model	Test Accuracy
Blank Models	47% - 63%
VGG16	97%
VGG19	92%

To understand the effects of the augmentation defined using the Keras `image_generator` function, an experiment was performed comparing the performance of the model with and without augmentation. Table 7 shows the results of this experiment.

**Table 7.** The importance of using image augmentation to improve test set prediction accuracy.

Augmentation	Test Accuracy
Yes	100%
No	43%

## 4 Results

Using a combination of CLAHE pre-processing and a fine-tuned VGG16-based CNN model, we have achieved 100% accuracy on a hold-out, cross-validation test set, using randomly selected images of the original data set. The model addresses the concerns of right-/left- handedness through image augmentation using the Keras augmentation. The output of the real-time image capturing, and reporting system web app provides lower accuracy due to the influence of back ground noise and image segmentation, however an assessment of the overall accuracy was not obtained.

## 5 Conclusions

In this work, we demonstrate a web-based ASL translation system based on Convolutional Neural Networks (CNN), that provides a practical application for real-time translation of 29 ASL signs, including 26 alphabet signs three additional signs ('space', 'delete', and 'nothing'). The keys to achieving this high precision in automated sign translation are 1) good input images (accomplished via CLAHE preprocessing) 2) starting from a pre-trained model (VGG16) and 3) fine-tuning of the model to the image library (Kaggle alphabet, digits, and custom hand signs).

Interrogation of the model via experimentation demonstrated that the model right-handed bias was prevented using the Keras' `image_generator`'s horizontal flipping option.



## 6 Future Work

This project focused on the first step of the fully-envisioned system. To complete system, the next step would develop NLP and NLU algorithms for ASL and English, or perhaps a dual-language corpus. Improvements in the CNN model could be explored via hyper-parameter optimization, such as evaluating the performance of GeLU as an activation function compared to ReLU.

The fully-envisioned system would be able to translate signs that incorporate movement. To accomplish this, the image capturing system would need to be improved to enable the capturing of video images, and the model would need to be re-trained, likely using a recurrent neural network (RNN), to address the time sequencing and ordering of the image data. Other system improvements, such as ensuring that genetic biases are fully addressed, and decreasing the system's sensitivity to background noise and lighting conditions other areas for improvement.

## 7 Program Code

Python code and Data sets can be found at: <https://github.com/kjprice/smu-capstone>

## References

1. Keras Documentation. Retrieved from: <https://keras.io/applications/#vgg16>
2. World Health Organization Fact Sheet. (2018, March). Deafness and Hearing Loss. Retrieved Oct, 2018 from <http://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>
3. Research Support & International Affairs (2011). Retrieved from <https://research.gallaudet.edu/Demographics/deaf-US.php>
4. History of American Sign Language. Retrieved from <https://www.startasl.com/history-of-american-sign-language>
5. KinTrans, Dallas, TX, US. url: <http://www.kintrans.com/index.html#product>
6. Geng1, Xin Ma, Xue, Wu, Gu, Lil: Combining Features for Chinese Sign Language Recognition. IEEE 2014. 11th IEEE International Conference on Control & Automation. Retrieved from: <https://ieeexplore.ieee.org/document/6871127>
7. Izzah, Abidatul & Suciati, Nanik. (2014). Translation of Sign Language Using Generic Fourier Descriptor and Nearest Neighbour. International Journal of Cybernetics and Informatics (IJCI). 3. 31-41. 10.5121/ijci.2014.3104. Retrieved from: [https://www.researchgate.net/publication/260294537\\_Translation\\_of\\_Sign\\_Language\\_Using\\_Generic\\_Fourier\\_Descriptor\\_and\\_Nearest\\_Neighbour](https://www.researchgate.net/publication/260294537_Translation_of_Sign_Language_Using_Generic_Fourier_Descriptor_and_Nearest_Neighbour)
8. Vi N. T. Truong ; Chuan-Kai Yang ; Quoc-Viet Tran (2016). A translator for American sign language to text and speech. 2016 IEEE 5th Global Conference on Consumer Electronics.

9. O. Koller, H. Ney, and R. Bowden. Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data Is Continuous and Weakly Labelled. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3793-3802, Las Vegas, NV, USA, June 2016. Retrieved from: <https://ieeexplore.ieee.org/document/7780781>
10. Kaggle hand sign database. url: <https://www.kaggle.com/grassknotted/asl-alpha-bet>
11. Wikipedia- Adaptive Histogram Equalization. Obtained from url: [https://en.wikipedia.org/wiki/Adaptive\\_histogram\\_equalization](https://en.wikipedia.org/wiki/Adaptive_histogram_equalization)
12. Singh, Arul Krishna, Kang, Pankaj. TheAILearner, 2018. Retrieved from: <https://theailearner.com/2019/04/14/adaptive-histogram-equalization-ahel/>
13. OpenCV tutorial on histogram equalization. url: [https://docs.opencv.org/3.1.0/d5/daf/tutorial\\_py\\_histogram\\_equalization.html](https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html)
14. Stanford Computer Science Department CS231 GitHub. Retrieved October 2018 from <http://cs231n.github.io/convolutional-networks/#norm>
15. Keras: The Python Deep Learning Library. url: <https://keras.io/>
16. TensorFlow. url: <https://www.tensorflow.org/>
17. Theano. url: <http://deeplearning.net/software/theano/>
18. CNTK. <https://github.com/Microsoft/CNTK>
19. Chollet, Francois and others , “Keras”, url: <https://keras.io>, 2015
20. The Theano Development Team, “Theano: A python framework for fast computation of mathematical expressions”, arVix:1605.02688v1, May 2016, url: <http://www.deeplearning.net/software/theano/>
21. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems”, 2015. Retrieved from: <https://arxiv.org/abs/1603.04467>
22. Amazon EC2: url: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
23. Colaboratory. URL: <https://colab.research.google.com/notebooks/welcome.ipynb>
24. Simonyan, Karen, Zisserman, Andrew, “Very Deep Convolutional Networks for Large-Scale Image Recognition” url: <https://arxiv.org/abs/1409.1556>, 2014
25. Gopalakrishnan, Kasthurirangan & Khaitan, S.K. & Choudhary, Alok & Agrawal, Ankit. (2017). Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. Construction and Building Materials. 157. 322-330. 10.1016/j.conbuildmat.2017.09.110. url: [https://www.researchgate.net/publication/319952138\\_Deep\\_Convolutional\\_Neural\\_Networks\\_with\\_transfer\\_learning\\_for\\_computer\\_vision-based\\_data-driven\\_pavement\\_distress\\_detection](https://www.researchgate.net/publication/319952138_Deep_Convolutional_Neural_Networks_with_transfer_learning_for_computer_vision-based_data-driven_pavement_distress_detection)

26. Dewinta & Heryadi, Yaya. (2015). American Sign Language-Based Finger-spelling Recognition using k-Nearest Neighbours Classifier. 10.1109/ICoICT.2015.7231481. Retrieved from: [https://www.researchgate.net/publication/279198249\\_American\\_Sign\\_Language-Based\\_Finger-spelling\\_Recognition\\_using\\_k-Nearest\\_Neighbours\\_Classifier](https://www.researchgate.net/publication/279198249_American_Sign_Language-Based_Finger-spelling_Recognition_using_k-Nearest_Neighbours_Classifier)
27. S. S. Fels and G. E. Hinton, "Glove-talk: a neural network interface between a data-glove and a speech synthesizer," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 2–8, 1993. Retrieved from: <https://www.cs.toronto.edu/~hinton/absps/glove-talki.pdf>
28. Savur, Celal. American Sign Language Recognition System by Using Surface EMG Signal. Rochester Institute of Technology. RIT ScholarWorks. December 2015 Retrieved from: <https://drive.google.com/file/d/17AH07N0zIDb5w7RpU7U64B98wk3GxIEW/view>
29. R.-H. Liang and M. Ouhyoung, "A Real-time Continuous Alphabetic Sign Language to Speech Conversion VR System," *Computer Graphics Forum*, vol. 14, no. 3, pp. 67–76, 1995. Retrieved from: [https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.1995.cgf143\\_0067.x](https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.1995.cgf143_0067.x)
30. 13. S. A. K. Mehdi Y. N., "Sign language recognition using sensor gloves," *Proceedings of the 9th International Conference on Neural Information Processing*, vol. 5, pp. 2204–2206, 2002. Retrieved from: <https://ieeexplore.ieee.org/document/1201884>
31. Viola and Jones, "Robust real-time object detection". Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sampling. Vancouver, Canada, July 13, 2001. Retrieved from url: <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-IJCV-01.pdf>
32. Thad Starner, Joshua Weaver, and Alex Pentland, "A Wearable Computer Based American Sign Language Recognizer", retrieved from: <https://www.cc.gatech.edu/fac/Thad.Starner/p/journal/wearable-computing-based-asl-recognizer.pdf>
33. Otiniano-Rodríguez, G. Camara-Chavez and D. Menotti, Hu and Zernike moments for sign language recognition, in 2012 Int. Conf. Image Processing, Computer Vision, and Pattern Recognition (ICCV'12), 16–19 July 2012, Las Vegas, USA, pp. 1–5.
34. Guardino, Caroline & Chuan, Ching-Hua & Regina, Eric. (2014). American Sign Language Recognition Using Leap Motion Sensor. 10.1109/ICMLA.2014.110.
35. K. S. Raut, "Recognition of American Sign Language Using LBG Vector Quantization," 2014 International Conference on Computer Communication and Informatics ICCCI -2014), pp. 2–6, 2014.
36. Sole, M.M., & Tsoeu, M.S. (2011). Sign language recognition using the Extreme Learning Machine. *IEEE Africon '11*, 1-6. Retrieved from: <https://www.semanticscholar.org/paper/Sign-language-recognition-using-the-Extreme-Machine-Sole-Tsoeu/ce6b3d30d3a610376e0d64258baf474577bf5d7f#citing-papers>

37. C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun. Real Time Hand Pose Estimation using Depth Sensors. 2011 Computer Vision Workshops, pp. 1228-1234, 2011.
38. Krupali Suresh Raut ; Shital Mali ; Sudeep D. Thepade ; Shrikant P. Sanas. “Recognition of American sign language using LBG vector quantization”, 2014 International Conference on Computer Communication and Informatics, 2014, Referenced from; <https://ieeexplore.ieee.org/document/6921745>
39. OpenCV -url: <https://opencv.org/>