DeepLetters:
A Convolutional Neural Network Long Short-Term Memory (CNN-LSTM) Approach to Fingerspelling Translation

Introduction

Currently, over 466 million people in the world have disabling hearing loss, and this number is projected to reach over 900 million by 2050 (Deafness and Hearing Loss, 2019). On a daily basis the deaf must communicate with those who don't speak American Sign Language (ASL). With increasing growth of the hearing-impaired community, it is important to be able to improve the communication between the deaf and hearing communities. Currently, communicating between hearing and hearing-impaired requires a human translator be present at all times, which is inefficient and costly. An ASL translator built on Artificial Intelligence (AI) architecture could provide a cost effective approach freeing the hearing-impaired to interact more freely in public and social settings.

Translating full ASL is very complicated for AI because there is a variety of signs, similarities among signs with different meanings, a reliance on facial expressions during translation, and differences in the way people look and sign. Fingerspelling, however, can be considered less complicated because there are a limited number of possibilities and facial expressions do not really play a role in the identification of the letter.

Early attempts at using machine learning to build an AI translator were based on the Hidden Markov Model (HHM) and showed great promise. Liwicki & Everingham (2009) used an HHM to achieve a single letter accuracy of 84.1%, and a word accuracy rate of 98.9%. However, the train and test set were both signed by the same inexperienced signer against the same plain background, and required a signer-specific skin model, limiting the generalizability of their model outside of the dataset. Ricco & Tomasi (2010) were able to achieve an accuracy of 57.32% without a restricted dictionary for the HHM model and 92.68% when the HHM model was supplied a dictionary. The data was also composed of only a single signer with the same plain background which again would limit the generalizability of the model outside of their dataset. Depth images have been used to help translate fingerspelling (Pugeault & Bowden, 2011; Dong et al, 2015), but this approach requires a special camera (think Xbox Kinect) which creates an extra barrier to fingerspelling translation. Other approaches have used special gloves which have achieved very good results, but also suffer from the problem of creating extra barriers to translation (Oz et al, 2011). More recently, neural networks have found great success in the area of computer vision.

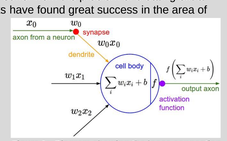


Figure 1: Diagram showing the loose connection to the brain and the equation for each neuron where w_i is the weight, X, is the input, and b is the bias for the neuron. (http://cs231n.github.io/neural-networks-1/)

Neural networks are a type of machine learning algorithm loosely based on the brain and are made of "neurons" that are assigned specific biases and weights (Fig. 1). The weights are multiplied by the inputs, a bias is added, and this product is fed through an activation function. An activation function is just a non-linearity that lets the network represent more complex ideas. Neural networks are trained using gradient descent, which is an iterative algorithm used to find the minimum of a function. It does so by finding where the gradient is largest and iteratively follows in that direction - the negative gradient - until this gradient becomes smaller. Gradient descent will always lead to a local minimum, as opposed to a global minimum in sufficiently large enough problems because the search space is astronomically large. Neural networks have been shown to effectively classify images. As a result, this technology has achieved widespread adoption in the field of image classification. Through image classification and artificial intelligence, there is opportunity to build a tool that can translate fingerspelling to text.

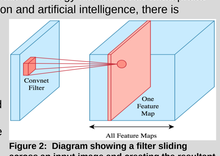


Figure 2: Diagram showing a filter sliding across an input image and creating the resultant feature map. (https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2)

A Convolutional Neural Network (CNN) is a type of neural network that is specifically made to handle image data. A CNN is different than a normal neural network because rather than assigning a different weight to every neuron, a filter, which can be thought of as a "sliding window" of weights is applied (Fig. 2). This means that instead of each input pixel having one weight assigned to it (like in traditional neural networks), a weight matrix of smaller size is slid over the image and used multiple times. This increases the network's ability to generalize because each filter can learn to look for a specific thing in the image. For example the filters early in the network may look for things like lines or curves, but filters further upstream may be looking for more complex things like fingers. These filters also significantly reduce the number of parameters that need to be trained, since every input no longer needs an individual weight assigned to it. These sliding matrices are 3-dimensional volumes, so each CNN layer has neurons in three dimensions. This is because images also have 3 dimensions: X, Y, and color (which, in computers, is represented by 3 RGB values).

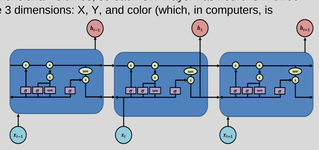


Figure 3: Shows multiple LSTM cells with their 4 gates. Gates that use the sigmoid function are denoted with σ. The gate that uses tanh is donated with tanh. Cell state is denoted by h sub t)

Garcia & Viesca (2016) used a CNN and achieved a single letter accuracy of 72% with static images of all letters (except J and Z). These results, however, did not transfer to real-world generalizability, as the translator was inaccurate outside of the training and testing data. Garcia and Viesca (2016) also attempted full-word translation, but used static images for each letter, telling the program when to move onto the next letter limiting its real-time application.

In order for real-time word translation to occur, an AI translator requires the ability to handle time-sequence data. Recurrent Neural Networks (RNN) have shown great promise in dealing with these time-sequence problems. RNNs have been used for text generation, image captioning, handwriting recognition, music generation, and language translation. LSTMs, which are a type of RNN, are typically recognized as being better than a traditional RNN because traditional RNNs have cells that only contain one gate, while an LSTM has four different gates in each cell (Fig. 3). Each gate is a neural network layer and has a specific purpose. One important issue LSTMs solve is the vanishing and exploding gradient problem, allowing longer sequences to be used more effectively.

A combined CNN-Long Short Term Memory (LSTM) model was attempted, trying both a encoder-decoder LSTM and a Connectionist Temporal Classification (CTC)-LSTM both with and without an R-Faster CNN which created bounding boxes around the hands (Shi, et al, 2018). Their highest accuracy was 41.9% using the CTC-LSTM with the R-Faster CNN.

The proposed method for this study uses a CNN-LSTM (Donahue, et al, 2015) combining the image discriminatory power of CNNs and the ability of LSTM's to handle time sequence data. Images or individual video frames are inputted into CNNs and the subsequent feature vectors are given to the LSTM cell for each time step (Fig. 4).

Figure 4: Shows method by which each frame of a video passes through a CNN and into the LSTM framework. The gates are represented by the purple boxes. Adapted from Karpathy (2017)

Data

CNN Data

The CNN data can be thought of as three groups: train, validation, and test. Training data are the data that the model will learn from and is what gradient decent uses to optimizes the network. Validation data are data that the network is not trained on and is used to evaluate the networking during training so that the network settings (hyperparameters) can be adjusted. Test data are similar to validation data, but unlike validation data are only run once after a best model is chosen from the validation scores. This is to avoid optimizing hyperparameters based on this test set to avoid overfitting and ensure a truly objective evaluation of the model.
The training and validation datasets came from four sources (Fig. 5):

- The University of Surrey's Center for Vision, Speech and Signal Processing is comprised of 65,774 images captured from 5 signers (Pugeault and Bowden, 2011). The image sizes from this dataset vary but average about 150x150 pixels. Images from one signer were selected for use as the validation set.
- GitHub dataset uploaded by a GitHub user which contains 7 signers and 1,680 images (Sreehari, 2016).
- The Massey University Gesture Dataset (Barczak, et al. 2011) contains 2,515, 500x500 pixel, up close hand images with a black background from 1 signer.
- Koller et. al (2016) dataset contained 2,288 garbage frames from 4 signers. Garbage frames do not contain any letter and include transitions between letters and no hands in the image at all. Because people are not always signing a letter, it is important for the model (especially the LSTM) to be able to tell the difference.

The testing data are comprised of an original dataset collected from the students at the Finalists' high school (Fig. 5). This dataset is composed of 332 images from two 3rd year ASL students, the ASL teacher, and the two Finalists themselves. Roughly half of the images were taken inside and the other half were taken outside.

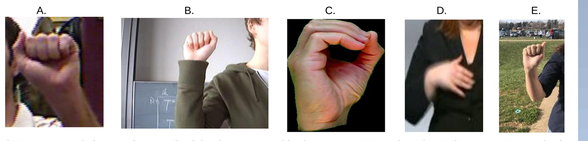
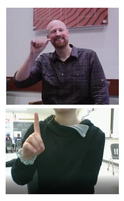


Figure 5: Example images from each of the datasets used in the CNN A - Pugeault and Bowden. 2011, B - Sreehari, 2016, C - Barczak, et al. 2011, D - Kollar, et al, 2016, E - High School ASL Dataset. These five images show the diversity of the dataset. Some images are zoomed in where the hand is, others have different lighting and show the entire arm, others are in outside environments, and some have black backgrounds to ensure only the hand is visible. The vast majority of the dataset is like A which comprises of 94% of the data

LSTM Data

The LSTM data are videos of people fingerspelling words and contains the following datasets:

- The Shi et al (2018) dataset contains 7304 videos from 3 signers in real world environments, and are self described as a challenging dataset. It contains some special characters (, &, ', .,@), proper nouns, other words, and abbreviations.
- The High School ASL dataset contains 250 videos from 5 signers.
- The HandSpeak (2019) dataset contains 267 videos created by 4 signers.



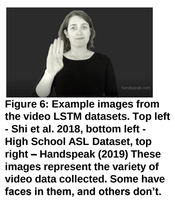


Figure 6: Example images from the video LSTM datasets. Top left - Shi et al. 2018, bottom left - High School ASL Dataset, top right – Handspeak (2019) These images represent the variety of video data collected. Some have faces in them, and others don't.

CNN

CNN Methodology

Pre-trained model

The CNN model used in this research (DeepLetters) started with the Deep Hand model (Kollar et al., 2016) and was retrained on ASL fingerspelling static hand images, and will act as the character-based model. The DeepHand model was used because it was trained on similar data (hand shapes that were not ASL fingerspelling static images). The Deep Hand model was derived by a retraining of the pre-trained GoogLeNet (Szegedy, et al., 2015).

Preprocessing

The images were resized so that the aspect ratio was maintained and they were padded to reach 256x256 pixels. From this, random crops of 227x227 pixels were then taken. Images were then normalized by subtracting the mean image which was also composed of random crops of 227x227 pixels from each image. Images were also randomly flipped to simulate both left and right signers.

Hyperparameters

Learning rates from 0.01 to 1e-7 were tried in combination with Adam (Kingma and Ba, 2014), and the gradient flowed to all layers. The parameters were tested by training a model and manually stopping the training when learning plateaued.

Garbage Frames

The base model was originally trained and validated on just images containing ASL fingerspelling signs. One issue that arises when every output is a letter is that even when there is no letter the model will try and predict one. This could be a problem when trying to translate entire words because there are "garbage frames" that are just the transitions between letters. Because of this, the enhanced model was trained with the addition of an extra output category for garbage frames.

CNN Results

Base Model (Without Garbage Frames)

Table 1: Percentages showing how various learning rates and regularization affected the validation accuracy and top 5 accuracy of the model. At a learning rate of 1e-4 →* 1e-6 and no regularization, the highest accuracy was achieved at 89.76%. At a learning rate of 1e-6, the highest top 5 accuracy was achieved at 97.80%
*** The arrow indicates that the learning rate was changed part way through the training**

Learning Rate	Regularization	Accuracy	Top 5 Accuracy
1e-4 → 1e-6	None	89.76%	97.01%
1e-5	None	82.02%	95.92%
1e-6	None	76.75%	97.80%
1e-4 → 1e-6	L2	87.73%	96.39%

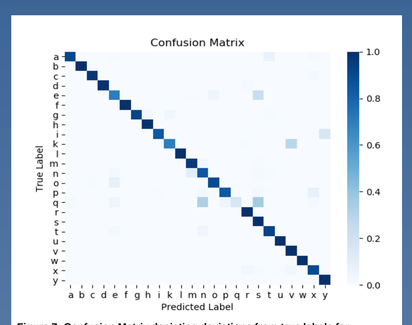
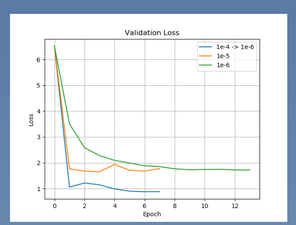
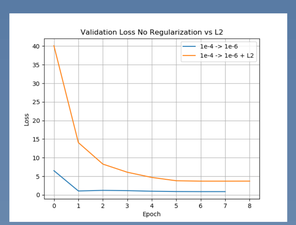


Figure 7: Confusion Matrix depicting deviations from true labels for fingerspelled letters. The model most notably confused q with s and n (darker squares not on the diagonal).

Validation Results

The base model CNN was able to reach about a 90% accuracy score obtained by training with a learning rate of 1e-4 → 1e-6 (Table 1). The top 5 accuracy values depict when the true label of the letter correctly shows up in the model's top 5 predictions. The model with the 1e-4 → 1e-6 learning rate obtained a 97.01% top 5 accuracy. However, the highest obtained was with the 1e-6 learning rate, achieving a 97.80% top 5 accuracy. The model was able to differentiate a majority of the letters with the exception of q (Fig. 7). The loss rates show that most of the learning occurred by at least the 4th epoch (Fig. 8).





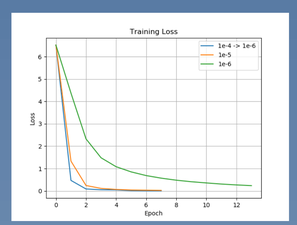


Figure 8:
Left –The validation loss for 3 learning rates. This graph shows that a majority of the learning stops after the first epoch, which can indicate that the learning rate is too high, but smaller learning rates got worse accuracies and higher loss values.
Middle – Compares the validation loss between the model with and without L2 regularization. With L2 regularization learning continues to happen after the first epoch, but it doesn't end at a better accuracy.
Right – The training loss showing that the amount of learning decreases drastically after the first epoch for the higher learning rates.

Test Set With Best Model

Accuracy
46.99%
Top - 5 Accuracy
75.60%

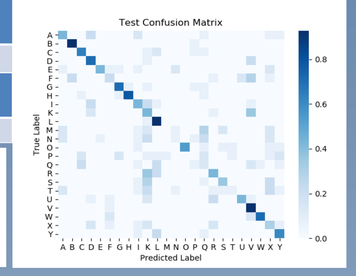


Figure 9: Confusion Matrix showing the misidentified signs in the test dataset. As can be seen by the greater number of squares not on the diagonal, this model had a low accuracy of 46.99%.

Results From Test Set

After testing the base model using the High School ASL dataset, the model was able to reach an accuracy score of 46.99%, lower than the accuracy from the validation set. The top 5 accuracy for the test set with the best model was 75.60%, which was a decrease from the validation set top 5 accuracies. The Confusion Matrix (Fig. 9) shows how the model was able to recognize B, L, and V with high accuracy (dark squares on the diagonal, approximately 80-100%). The letters Y, W, H, G, C, and D were also detected with reasonable accuracy (approximately 60-80%). Letters like M, N, P, Q, S, T, and U were much more difficult to detect from the test set (approximately 0-30%).

Enhanced Model (With Garbage Frames)

Table 2: Percentages showing how various learning rates and regularization affected the validation accuracy and top 5 accuracy of the model. At a learning rate of 1e-4 and no regularization, the highest accuracy was achieved at 93.27%. At a learning rate of 1e-6, the highest top 5 accuracy was achieved at 97.80%

Learning Rate	Regularization	Accuracy	Top 5 Accuracies
1e-4	None	93.27%	97.72%
1e-5	None	83.53%	97.07%
1e-6	None	79.61%	97.80%
1e-4	L2	86.91%	96.53%

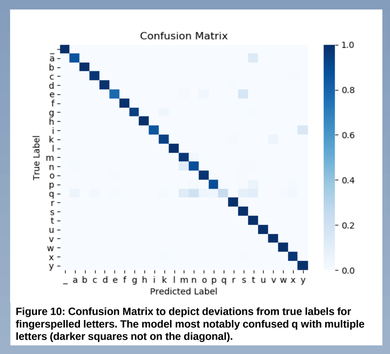
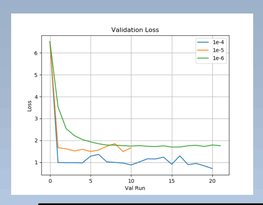
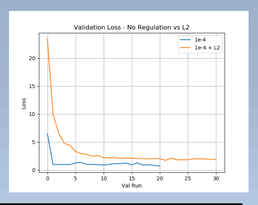


Figure 10: Confusion Matrix to depict deviations from true labels for fingerspelled letters. The model most notably confused q with multiple letters (darker squares not on the diagonal).

Validation Results

The enhanced model CNN was able to reach a 93% accuracy score obtained by training with a learning rate of 1e-4 (Table 2). The top 5 accuracies value depicts when the true label of the letter correctly shows up in the model's top 5 predictions, for which the 1e-6 learning rate obtained a 97.80% top 5 accuracy, and the 1e-4 learning rate achieved a 97.72% top 5 accuracy, which was marginally lower. The model was able to differentiate a majority of the letters with the exception of q (Fig. 10). The loss rates show that most of the learning occurred by at least the 4th epoch (Fig. 11).





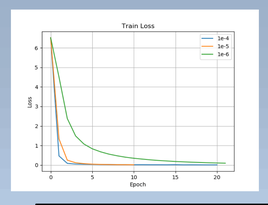


Figure 11:
Left – Validation loss for 3 learning rates. The model with 1e-4 shows the loss fluctuating considerably and eventually approaches a local minima.
Middle – Similarly with the L2 loss of the base model, more learning happens after the first epoch with the L2 model as compared to the model with the same learning rate without L2 regularization.
Right – Shows that the model is converging after one epoch for the higher learning rates, showing similar trends to the base model.

Test Set Results With Best Model

Accuracy
45.48%
Top - 5 Accuracy
75.90%

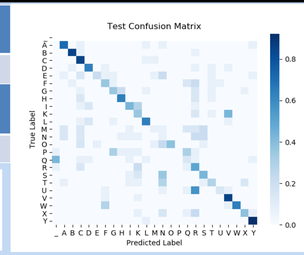


Figure 12: Confusion Matrix showing the misidentified signs in the test dataset. As can be seen by the greater number of squares not on the diagonal, this model had a low accuracy of 45.48%.

Results From Test Set

After testing the base model using the High School ASL dataset, the enhanced model was able to reach an accuracy score of 45.48%, lower than the accuracy from the validation set and lower than the base model test accuracy of 46.99% (Fig 12). The top 5 accuracy for the test set with best model was 75.90%, which was an increase from the base model validation set top 5 accuracies, which was 75.60%. The Confusion Matrix shows how the model was able to recognize B, C, V, and Y with high accuracy (approximately 80-100%). A, D, H, L, K, and W were also detected with reasonable accuracy (approximately 60-80%). Letters like M, N, P, Q, S, T, and U were still difficult to detect from the test set (approximately 0-30%). In comparison to the base model H, D, and W overlapped as letters detected with reasonable accuracy, but L moved from high accuracy to reasonable accuracy. B and V were still detected with high accuracy, but Y and C increased from reasonable to high accuracy.

LSTM

LSTM Methodology and Results

Based on the approach of Shi, et al (2018) Seq2Seq and CTC models were identified to be tried as the LSTM model. The difference between their approach and the approach laid out by this work is that the CNN being plugged into the LSTM has already been trained on fingerspelling images, which could help converge to a better solution. The dataset being used for the LSTM is very different from the dataset used to train the CNN. The CNN dataset was all of close-up, static hand images, while the LSTM dataset contains videos of people close, far, and using two hands. Because of this, the CNN in the new CNN-LSTM architecture must be trained again with the LSTM. However, this creates a large memory requirement for running the model. Initial attempts using this methodology revealed a need for greater computing resources than are currently available to the Finalists. Access to cloud computing space and time will be required to proceed with this step.

Discussion

It is promising that the DeepLetters CNN, by reaching an 89.76% accuracy (Table 1), was able to surpass the 72% validation accuracy bar that Garcia and Viesca (2016) set because both models' architecture were similar (both were a version of a Google inception net trained for the image net challenge). The methods used in training this network were also similar to those used by Garcia & Viesca (2016), using the same data they did with the addition of an extra dataset (Sreehari 2016). The DeepLetters CNN was able to converge to a higher accuracy because it was already trained on a million hand shape images by the DeepHand team (Kollar et al. 2016).

Pugeault & Bowden (2011) used a dataset consisting of depth images and reported an accuracy of 75% on their validation set, however, they reported no test sets. The DeepLetters CNN model, has a higher validation accuracy (89.76%) despite not using depth images. After adding the garbage frames to the base model, the validation accuracy was almost 3% higher than before (Table 2). This shows promise in the fact that the addition of garbage frames allowed for the model to throw out images that were blurred or did not contain a hand.

The test set accuracy was 47%, lower in comparison to the validation accuracies. When analyzing the confusion matrix, the letters M, N, P, Q, S, T, U, and F contained most of the error (Fig. 9). Part of this is likely because M, N, P, Q, S, and T are very similar in hand form. It is important to note that the diversity of signers in the dataset plays a big role in being able to detect letters like M, N, P, Q, S, and T with higher accuracy. There was trouble with F likely because some of the signers in the dataset signed F at an angle such that it was hard to tell the difference between F and V or U. Another reason for the inaccuracy could be that the images in the test dataset were taken from a bit further away in comparison to the images used for training.

The drop in accuracy for the test set between the base model and enhanced model was minimal, and this is encouraging because adding the garbage frames increase the complexity of the problem. With this increase in complexity, having only a 1.51% decrease (Fig. 9 and 12) in test accuracy shows the overall strength of the model. A method that could work to improve the garbage frame dataset would be to include hand sign images to the dataset that don't represent actual ASL fingerspelling signs (rather than the current dataset of images with transition frames or no hand at all). This could potentially force the model to really learn what identifies certain signs and discard any other handshape that doesn't represent a letter.

Testing models using robust, real-world test sets that are comprised of images from many signers in a variety of environments and captured separately from training and validation test sets is uncommon. Garcia and Viesca (2016) are a notable exception because their testing dataset consisted of images with a variety of signers and environments, however their models weren't able to generalize well. For example, one of their models classified each input as 'A" over half of the time with a confidence of over 99%. In contrast, the DeepLetters CNN model had an overall accuracy of 47%.

Unfortunately, it is difficult to compare the 47% test accuracy with other papers because of the limited use of real-world test sets. However, test set accuracy is still a valuable source of information that helps show how the DeepLetters model is capable of detecting some signs outside its own training and validation dataset. In order to reap the real-world benefits of fingerspelling and sign language translation using artificial intelligence, it is important to test models on real-world datasets with different signers and environments. The 47% accuracy of the DeepLetters CNN acts as a stepping stone toward future success in real-world translation.

Future Work

While the dataset used for the CNN was very large, most of it comprised only of 5 signers and was images of the hand very close up. Using an image dataset not dominated by a few signers, taken from a greater distance, taken in more locations, and in less-controlled environments will help improve generalizability.

Due to resource constraints, the LSTM portion of this work has yet to be completed. More computing resources is all that is required because the model itself is built and ready to be trained. Using a cloud-based approach would be a good option to get these required resources. When complete, results will be compared with Shi et al. (2018).

References

Barczak, A.L.C., Reyes, N.H., Abastillas, M., Piccio, A.,Susnjak, T. (2011), A new 2D static hand gesture colour image dataset for ASL gestures, ResearchLetters in the Information and Mathematical Sciences, 15, 12-20

Deafness and hearing loss. (2019). Retrieved on May 3, 2019 from https://www.who.int/news-room/fact sheets/detail/deafness-and-hearing-loss

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Darrell, T., & Saenko, K. (2015). Long-term recurrent convolutional networks for visual recognition and description. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2015.7298878

Dong, C., Leu, M. C., & Yin, Z. (2015). American Sign Language alphabet recognition using Microsoft Kinect. *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. doi:10.1109/cvprw.2015.7301347

Garcia, B., Viesca, S. A. (2016). Real-time American sign language recognition with convolutional neural networks. Retrieved from http://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf

Handspeak (2019). Retrieved November 1, 2019 from https://www.handspeak.com/spell/practice/

Karpathy, A. (2017). Convolutional Neural Networks (CNNs / ConvNets). (n.d.). Retrieved from http://cs231n.github.io/convolutional-networks/

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kollar, O., Ney, H., Bowden, R., (2016). Deep Hand: how to train a CNN on 1 million hand images when your data is continuous and weakly labelled. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Liwicki, S., & Everingham, M. (2009). Automatic recognition of fingerspelled words in British Sign Language (pp. 50–57). Retrieved from http://eprints.whiterose.ac.uk/8903/2/mark_everingham_2.pdf

Oz, C., & Leu, M. C. (2011). American Sign Language word recognition with a sensory glove using artificial neural networks. *Engineering Applications of Artificial Intelligence, 24*(7), 1204-1213. doi:10.1016/j.engappai.2011.06.015

Pugeault, N and Bowden, R (2011). Spelling it out: Real–time ASL fingerspelling recognition In: ICCV 2011: 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision, 2011, Barcelona, Spain.

Ricco, S., Tomasi C. (2010) Fingerspelling recognition through classification of letter to letter transitions. In: Zha H., Taniguchi R., Maybank S. (eds) Computer Vision – ACCV 2009. ACCV 2009.Lecture Notes in Computer Science, vol 5996.Springer, Berlin, Heidelberg

Shi, B., Rio, A. M., Keane, J., Michaux, J., Brentari, D., Shakhnarovich, G., & Livescu, K. (2018). American Sign Language fingerspelling recognition in the wild. *2018 IEEE Spoken Language Technology Workshop (SLT)*. doi:10.1109/slt.2018.8639639

Sreehari. (2016) Mon95 - Overview. Retrieved on November 1, 2018 from https://github.com/mon95

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2015.7298594

All images, tables, and figures were created by the Finalists unless otherwise indicated