CS230

Sign Language Recognition and Transcription with Neural Networks

Shubham Patil
Stanford University
shubhamp@stanford.edu
theshubhamp@gmail.com

Abstract

Sign Language helps people from the deaf community to connect with other people and communicate their thoughts. Although a fair number of people can communicate with sign language, inability to communicate with someone in times of emergency is a big problem. This report presents a Convolutional Neural Network trained on the Sign Language MNIST [1] Dataset with a Train and Test accuracy of 92%.

1 Introduction

American Sign Language or other variants enable the deaf community to share their thoughts with the world. Though there are people who can understand and communicate with sign language, a vast majority still has limited exposure to it. Services like Google Translate, Bing Translator etc. have played a huge role in reducing barriers to written and spoken communication. Research towards making Sign Language Recognition better would do the same for the deaf community and will technology more accessible and inclusive.

The model built as part of this project accepts an image or frame representing an American Sign Language Gesture, a convolutional neural network then processes it to generate a probability distribution of the possible matches as output.

Since Sign Recognition is a Computer Vision problem, challenges listed below usually affect the overall efficiency of the classification task:

- Lighting Conditions
- Identification of Region of Interests
- Sign Boundary Detection

A system capable of recognizing finger spellings and hand signs in a real-time stream, requires the following tasks to be performed:

- Object Localization to detect hand gestures from a sampled video stream.
- Recognizing a sequence of finger-spellings gestures as a word
- Recognizing a sequence of frames as words represented by motion signs.
- Constructing a sentence from the recognized words.

2 Related work

Use of Neural Networks for Sign Recognition is not a new approach, and prior work in the academia indicates that these researches can broadly be grouped under these two categories:

- Models trained with plain image data.
- Models trained with more sophisticated features like depth and positional coordinates in a 3D space.

A relevant study by Brandon Garcia and Sigberto Alarcon Viesca [2] explores the use transfer learning to be recognize static finger-spelling gestures. The methodology consisted of reusing a pre-trained GoogLeNet trained on the ImageNet - ILSVRC2012 and transfer the learnt parameters onto recognizing signs from the Surrey and Massey University Sign Language Datasets. The authors note that the limitation of the dataset affected their model's generalizability to signs for letters other than a - e.

Although not built atop neural networks, a different study by Rohit Sharma, Yash Nemani et al. [3] explores the use of image pre-processing tasks, like contour detection to recognize American Sign Language in face of inherent ambiguity in the language's vocabulary. I found this approach to be particularly novel because it attempted to extract more meaningful features from a existing datasets in an effort to improve efficiency. The approaches highlighted in the paper can be realized as a pre-processing pipeline and neural network model to achieve better generalization.

A different study by Lionel Pigou(B), Sander Dieleman et al. [4] although for Italian Gestures was able to achieve a cross-validation accuracy of 91.7%. Their approach to this model used Kinect for depth information. Their model was able to generalize well for unknown surroundings and users.

A similar study using a Custom CNN architecture was carried out by Vivek Bheda and N. Dianna Radpour [5]. The network had a fairly common architecture with a series of 3x3 Conv Layer Pairs followed by a Max Pool Layer and Dropout.

3 Dataset and Features

To build a baseline model to recognize finger-spellings, the "Sign Language MNIST" [1] Dataset is used. This dataset has the following characteristics:

- 24 target classes from representing letters A-Z except J and Z as they require motion
- 27455 & 7172 Samples for Training and Test.
- Each sample is a 28 x 28, single color channel image.

This dataset itself was built by the author by augmenting on a small number of images (1704) by cropping, rotating and adding noise.

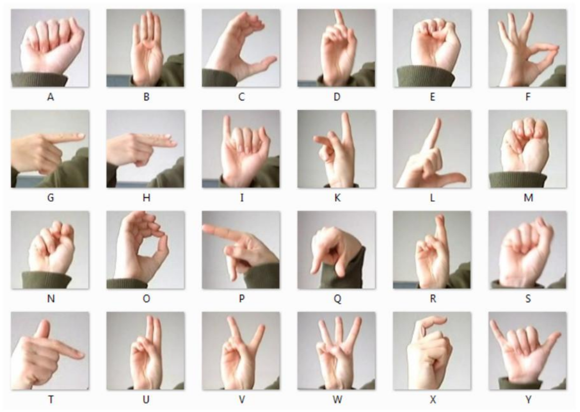

Figure 1: Letters in American Sign Language

4 Methods

Loss Function
Sparse Categorical Cross-Entropy was used as a loss function for this model. It is a variant of Categorical Cross-Entropy loss function which works if the target model output is a label value instead of a on-hot encoding.

This loss is also called a Softmax Loss and is represented as a combination of a Softmax Activation coupled with a Cross Entropy Loss and can be represented like this:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{e^{f_{i,y_i}}}{\sum_{j=1}^{C} e^{f_{i,j}}}\right) \tag{1}$$

$$f_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \tag{2}$$

N = total number of training examples
C = total number of classes

Equation 1 calculates the mean loss for each training example and Equation 2 is a SoftMax Function and accepts a vector as input to convert it into a vector of values [0, 1] which sum to 1. When clubbed together the output from this loss function can be used to train for multi-label classification problems.

Algorithm
Architectures with different combination of layers and Activation functions were tried, but the final finger-spelling recognizer was built using a convolutional network with 7 layers with this architecture:

- 2D Convolution with 64 filters with window size 3x3. ReLU Activation.
- 2D Max Pooling Layer with size 2x2.
- 2D Convolution with 64 filters with window size 3x3. ReLU Activation.
- Flattening Layer.
- Densely Connected Layer with 120 neurons. ReLU Activation
- Dropout Layer is 20% of input units to drop.
- Densely Connected Layer with 26 neurons. Softmax Activation, for 26 Clases A-Z

Libraries and Code
The model was built with code using Tensorflow 2.0 [6] and its Keras APIs. Slicing and dicing of data into train and test splits was done using NumPy [7] Reading and manipulation of data from cameras made use of the OpenCV Project [8] and it's python bindings.

5 Results and Things Learned

The model built as part of this project was able to achieve a training accuracy of 99.92% and a validation accuracy of 96.19% when run for 50 epochs. Although the model achieved promising performance numbers, it over-fit the training dataset and could not generalize well on examples and videos outside the dataset. This resulted in the transcription objective of this project to remain unexplored. The failure in generalizability can be attributed to the following reasons:

- Low Dimensional Nature of the Dataset.
- Dataset biased to a single ASL Signer.

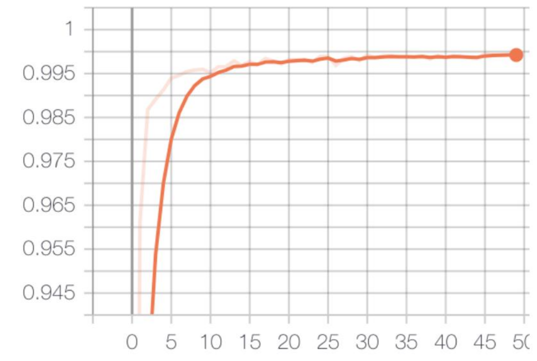

Figure 2: Epochs vs Accuracy in the Training Step

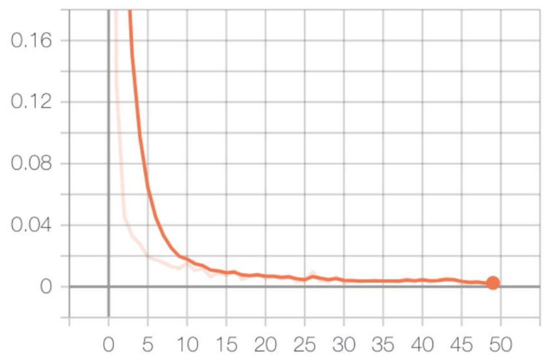

Figure 3: Epochs vs Loss in the Training Step

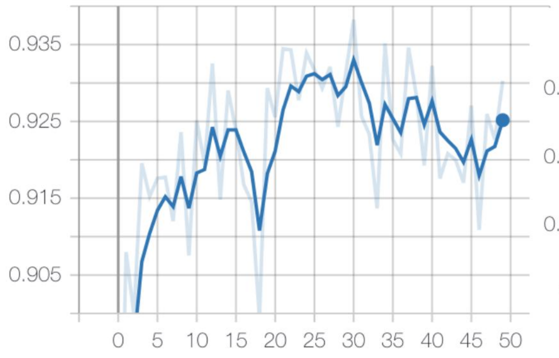

Figure 4: Epochs vs Accuracy in the Validation Step

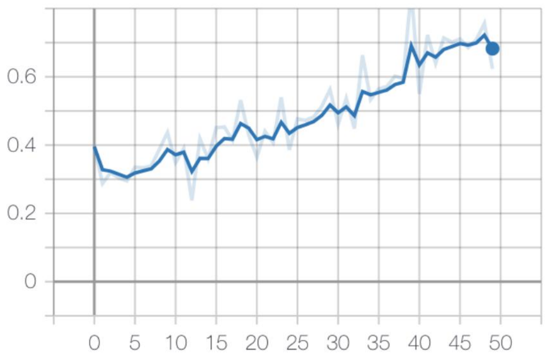

Figure 5: Epochs vs Loss in the Validation Step

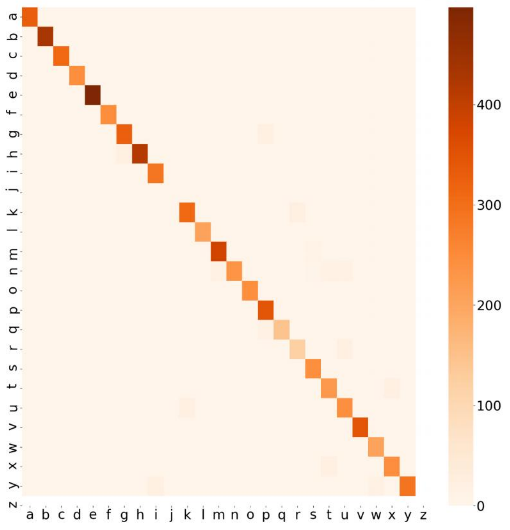

Figure 6: Confusion Matrix, Predicted Labels vs Actual Labels

6 Conclusion

Because of the nature of the dataset, the model was not able to generalize well on samples from outside the dataset like realtime camera feed. But looking at the encouraging results, I feel a more sophisticated data collection and augmentation process can help with this problem.

7 Contributions

This project was worked on by a single contributor: Shubham Patil.

References

[1] tecperson on Kaggle. "Sign Language MNIST" https://www.kaggle.com/datamunge/sign-language-mnist

[2] Garcia, Brandon, and Sigberto Alarcon Viesca. "Real-time american sign language recognition with convolutional neural networks." Convolutional Neural Networks for Visual Recognition 2 (2016).

[3] Sharma, Rohit, et al. "Recognition of single handed sign language gestures using contour tracing descriptor." Proceedings of the World Congress on Engineering. Vol. 2. 2013.

[4] Pigou, Lionel, et al. "Sign language recognition using convolutional neural networks." European Conference on Computer Vision. Springer, Cham, 2014.

[5] Bheda, Vivek, and Dianna Radpour. "Using deep convolutional networks for gesture recognition in American sign language." arXiv preprint arXiv:1710.06836 (2017).

[6] Google. "TensorFlow" An opensource framework for Machine Learning https://www.tensorflow.org

[7] NumPy Developers. "Numpy" A package for scientific computing with Python https://www.numpy.org

[8] OpenCV Team. "OpenCV" An open source computer vision and machine learning software library https://opencv.org

[9] skvark on GitHub, "opencv-python" Unofficial pre-built OpenCV packages for Python https://pypi.org/project/opencv-python