ASL Image Classification

By: Myra Rust

Executive Summary

This project uses machine learning techniques to test computer vision capabilities on human hand gesture images. Using a convolutional neural network to predict the letter value of the hand pose from a 2D image a model achieving an accuracy score of 92.1% was achieved. This is a significant accomplishment and could lead to further work incorporating machine learning models to provide real-time interpretation of sign language.

Introduction/Background

American Sign Language (ASL) is a fully developed natural language that uses movements of the hands and face to communicate. ASL is used throughout North America by persons who are deaf, hard of hearing, have other communication disorders, and many other persons. Each letter in the English language directly correlates to a hand pose in the ASL alphabet and words can be spelled out letter by letter.

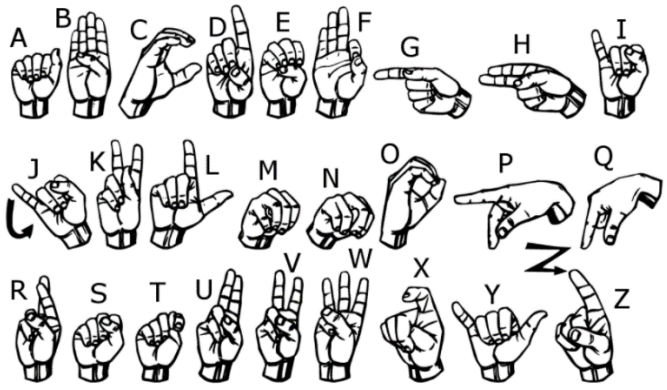


Figure 1: ASL Letters (Tecperson, 2017).

The objective of this project is to determine if machine learning can be used to properly classify static images of ASL hand poses. The results of this project could stand to be the foundation for further language translation of American Sign Language. The scope of this project focuses on static poses and letters, but if successful, with today's technology this could be expanded to possibly include hand and facial movement recognition of the entire ASL dictionary. From there, machine learning could be used to conduct real-time translation of ASL. How cool would it be for a person to have a virtual assistant in their home that they could interact with using ASL, just like we do with speech today.

Preliminary Analysis

Data Source: The training and test datasets used for this project were created using 1,704 original pictures of hand poses. Those images were cropped, gray-scaled, resized to 28x28, rotated, altered to have varying contrast, and pixel noise was added to ensure that no two images were the same. The training set contains 27,455 total images, and the test set contains 7,172 total images. These images are presented as a CSV containing 784 features representing the pixels and one feature 'label' that is a number corresponding with the letter in the alphabet that is represented. J and Z are excluded from this project as they require movement and are not static hand poses.

A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
0	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Figure 2: Letter to Target Value Mapping

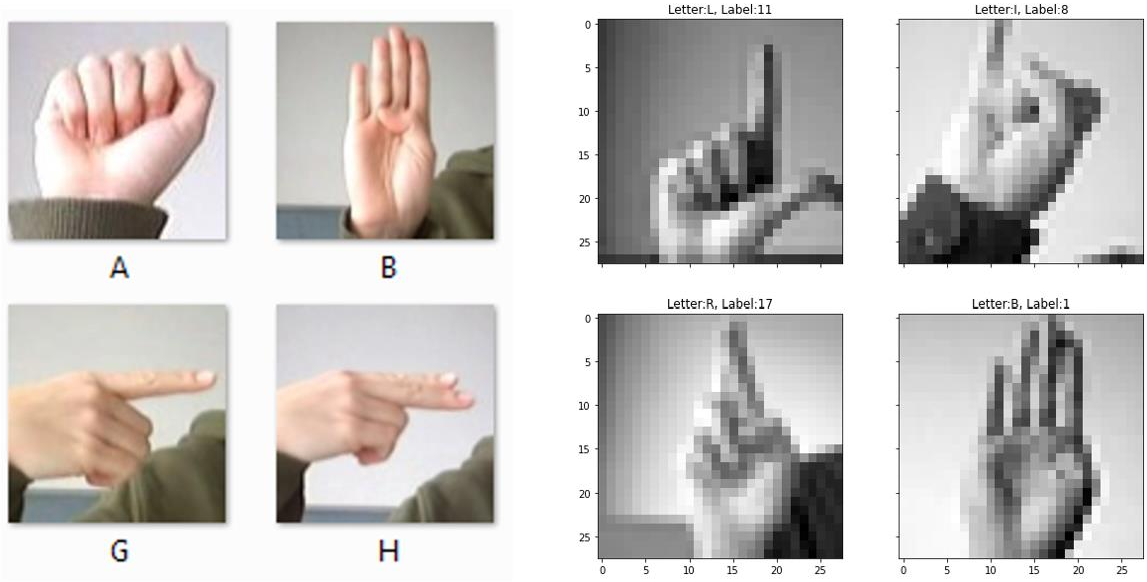


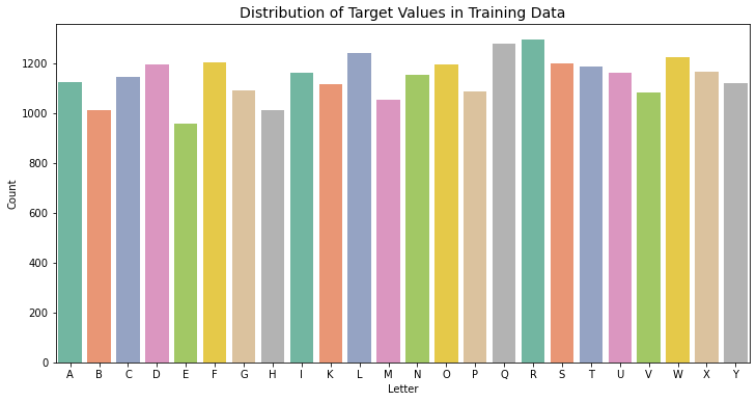
Figure 3: Pre and Post Processed Image Samples

To see a more comprehensive look at pre and post processing samples, refer to *Appendix A*: Pre and Post Processed Data.

Dataset URL: https://www.kaggle.com/datamunge/sign-language-mnist

Data Transformation:

The data provided was already processed and transformed by the dataset creator using a data augmentation pipeline. The training dataset provided contained images that were evenly distributed across the target label, which can be seen in the image below.



The data was already split into training and test datasets. I choose to further split the training dataset into training and validation sets. With the final shape being:

- Training dataset: 20,000 images
- Validation dataset: 7455 images
- Test dataset: 7172 images.

Model Selection and Evaluation

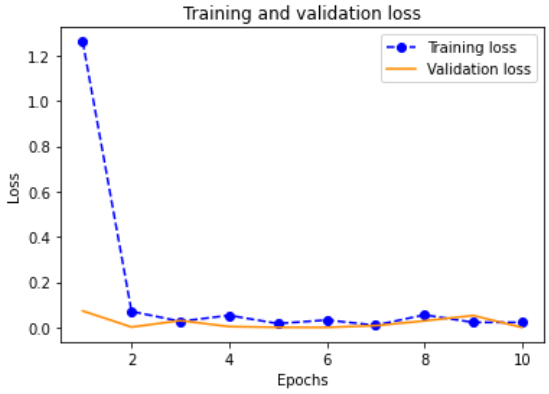
The objective of this project 'to properly classify hand pose images' is a problem of single-label multiclass classification. A Convolutional Neural Network (CNN) was selected to process the data because CNNs are commonly used to analyze imagery data. This was done leveraging the Keras library with a Tensorflow backend to create the neural network.

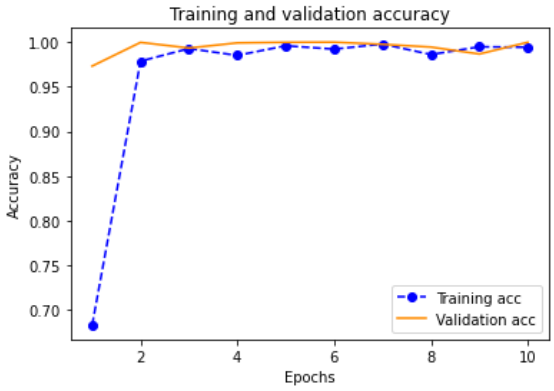
When constructing the layers of the neural network, layers were kept big enough to prevent information bottlenecking. The model included three Convolution2D layers and two MaxPooling layers. A dropout layer was added directly before the last Dense layer to mitigate overfitting. The softmax activation function was used in the last Dense layer to allow results to be separated into multiclass labels.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 128)       36992
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 512)         590336
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              4719616
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 25)                6425
=================================================================
Total params: 5,616,089
Trainable params: 5,616,089
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: CNN Model Summary

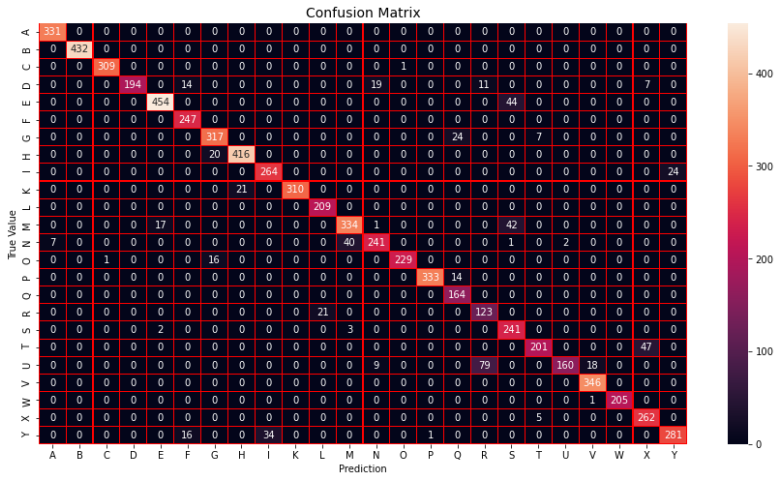
When compiling the model, the Keras sparse_categorical_crossentropy loss function was selected because the target variable is numeric, categorical, and not one-hot encoded. Optimizers are used to minimize loss by adjusting the learning rates of gradient descent. The model was then evaluated using both the Adam and RMSProp optimizer functions. The Adam optimizer function provided the best results. “Adam is a first-order-gradient-based algorithm of stochastic objective functions, based on adaptive estimates of lower-order moments.” (Kingma & Ba, 2014). Adam is known to provide great results with computer vision tasks. Below are the resulting training and validation plots of loss and accuracy. Overfitting seemed to occur very quickly with the validation set, therefore, a total of two epochs were chosen to use for the final model.





Results

The final model was trained using the entire training dataset. The model was then compiled from scratch and ran with the test dataset. The model achieved an accuracy score of 0.921. That means that out of every 100 predictions, ~92 of them were correct. We can use the confusion matrix to take a closer look at the predictions and identify where the model had difficulty.



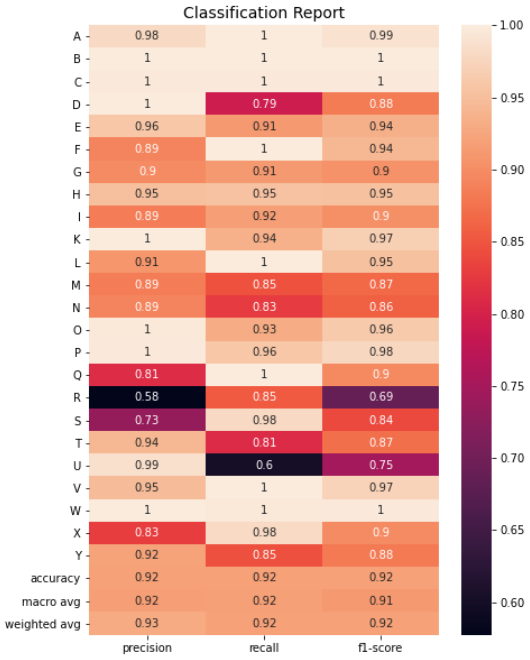
The top five mistakes the model made with the test dataset were:

- Predicting U as R: 79x
- Predicting T as X: 47x
- Predicting E as S: 44x
- Predicting M as S: 42x
- Predicting Y as I: 34 x

What is interesting with all these mistakes is the hand poses are very similar, please refer to *Appendix A*: Pre and Post Processed Data to view the hand gestures. One could ascertain why a computer viewing a 2D image broken down into pixel counts might interpret them incorrectly it's analogous with a human trying to interpret these hand gestures without there prescription glasses, I would venture to guess that humans would make similar mistakes as the model.

The confusion matrix doesn't provide a complete picture though because observation counts varied between features, so we can view the results in percentages of precision, recall, and F1 score in the classification report below. Precision is the positive predictive power and can best be explained as the percentage of predicted values that matched the true value for each letter.

Recall is the true positive rate or sensitivity and can best be explained as the percentage of true values predicted correctly for each letter. F1 score uses both the precision and recall percentages to calculate a single measurement of model accuracy. We can see as a whole that the top three letters giving out model difficulty were R, U, and S.



Conclusion

In conclusion, an accuracy score of 92.1% is a successful outcome for this project. This prediction model proves that computer vision can be used to accurately interpret ASL hand poses, and this knowledge can be used as a foundation going forward to employ convolutional neural networks in the interpretation of moving hand gestures.

Future Work

Even though this project was a success, there is some potential future work that could be done in attempt to further improve the accuracy score of the model. This includes adding more images to the training and test sets. Images that are not cropped so closely to the hand, that

contain different lighting, and that are from persons of differing skin tones might help to provide better training for the model.

In addition to improving our model accuracy with future work, it would be exciting to expand this work to contain not just letter hand poses, but word poses or moving gestures for the entire ASL dictionary and associated facial gestures.

Acknowledgements

I would like to thank Professor Williams and all other professors that are part of Bellevue University's data science program. Without their direction, support, and guidance, this paper would not have been possible. I would also like to thank my daughter for sacrificing her time with me to allow me to pursue this degree for the betterment of myself.

References

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. ICLR 2015. Retrieved from https://arxiv.org/pdf/1412.6980.pdf.

Tecperson. (2017, October 20). Sign Language MNIST. Version 1. Retrieved on September 28, 2021 from https://www.kaggle.com/datamunge/sign-language-mnist.

Wikipedia contributors. (2021, September 27). American Sign Language. In *Wikipedia, The Free Encyclopedia*. Retrieved 01:08, October 1, 2021, from https://en.wikipedia.org/w/index.php?title=American_Sign_Language&oldid=1046848268

Appendix A: Pre and Post Processed Data

Figure 5: Preprocessed data (Tecperson, 2017).

Figure 6: Postprocessed data (Tecperson, 2017).