Detecting the Sign Language Using Neural Network.

With Images of American Sign Language (ASL)

SEOK HWAN SONG

Iowa State University

ABSTRACT

Neural network is being used for several fields, and especially for image processing it is very broadly being applied. With the images of American Sign Language, the network has been used to detect people's sign languages. Convolution neural networks are applied to detect the images, and some preprocessing and methods are announced and conclusions and future works are also discussed. The images of ASL are able to be detected very well with CNN and would be available to be extended to other applications in the future.

CCS CONCEPTS • Neural Network, Convolutional Neural Networks (CNNs)

Additional Keywords and Phrases: Sign Language, American Sign Language, PyTorch

1 INTRODUCTION

Approximately 1,000,000 people in the U.S. are 'functionally deaf' among people over 5 years old, and more than 50% of people over 65 years old are 'functionally deaf as well. Moreover, approximately 600,000 people in the U.S. are 'deaf' and about 2.2% of American people are having troubles with hearing [2]. Including the number of mutes, in the world there are around '700,000 to 900,000 of deaf-mutes [3]. Therefore, we can say that there are so many people are using sign languages to communicate. However, not many people are able to use the language and most of applications for languages are for actual languages. For example, translating applications only detects the sounds of people speaking. Most of apps related to sign languages are for educational methods and as long as the language is 'a visual language', applications take so much images and videos [4]. However, sign languages are still languages and people need to develop many functions for people using the languages. As people detect other languages with sounds and letters, we can detect sign languages with videos and images.

With artificial intelligence, all process can be helped and the accuracy can be increased as well. To train a model and detect American Sign Language (ASL), sign language MNIST dataset is used and a neural network model is applied and some methods are conducted to evaluate the results.

2 METHODOLOGY

To train and test the dataset, data preprocess, network models, and simulation methods are conducted, and PyTorch is used through the whole process. The detailed process will be discussed in this section.

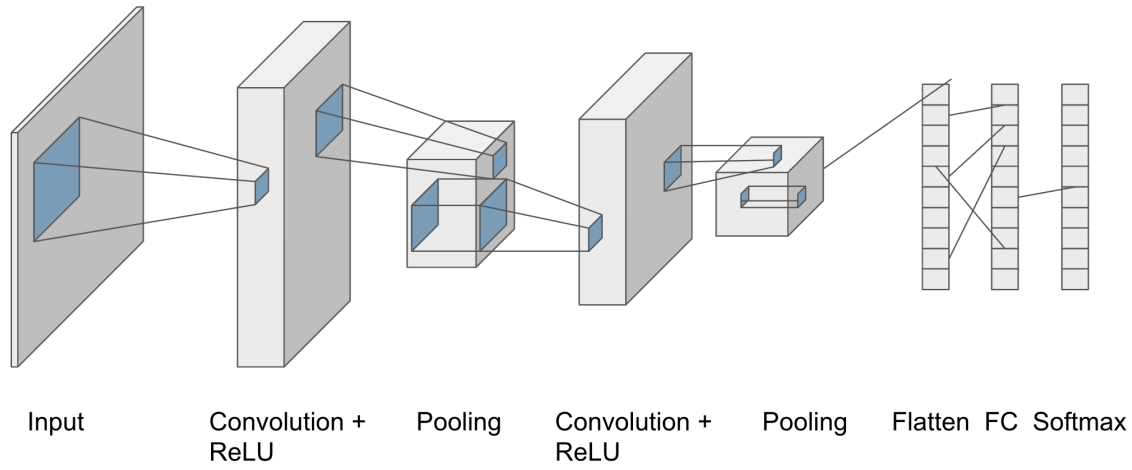
2.1 Data Preprocess

The dataset of 'Sign Language MNIST' is used [1]. As long as, network with PyTorch is applied, datasets should be converted into a certain format. After the dataset is read, it is transformed with several steps: random horizontal flip, color jitter and normalize. Loading dataset is the next step to apply for PyTorch models. Training dataset and testing dataset after being transformed are loaded with batch size 4. During the loading data, datasets are shuffled and reshaped as well.

2.2 Model

The model has several layers. At the beginning, 'three layers convolution neural network with batch size 32' is applied and the accuracy of the first stage is 71%. Therefore, to increase the accuracy, the batch size 4 and different neural network is conducted and the accuracy goes up to 83.11% on the first round. Therefore, the model is used and the model is explained on the Figure 1. When input goes into the model, a convolution and a ReLU layer will deal with the input image data and then send it to a pooling layer. After that, one more convolution, ReLU, and Pooling layer will be applied. When the layers are done, it will be flattened and fully connected layer and SoftMax is used to get the predicted label. The sizes of tensors are small for the model, so the accuracy goes up when the sizes are increased. However, that takes much more time to train with the bigger sizes, so the small sizes are applied. It ends with accuracy 93.10% at the end and it takes 850 seconds.

Figure 1: Styles available in the Word template



3 SIMULATION RESULTS

Through this part, results from the simulations are going to be announced and explained. Some tables will be used to evaluate the model.

3.1 Tables

Table 1 is a confusion matrix about the trained model. It compares the true labels and predicted labels. It has some wrongs but we can say that the precision of the model is very high with the dark colors on the orthogonal of the matrix. For the wrongs, especial 4 and 17 have high number of false predicted images and 17 is being confusing with 20 as well, and 18 and 22 have very large number as well. 4 is E, 17 is R, 20 is U, 18 is S, and 22 is V. With Figure 2, we can say that R and U are very similar and E and R, and S and V can be confusing as well. However, the accuracy is still very high and this confusion can be improved more.

Table 1: Confusion Matrix

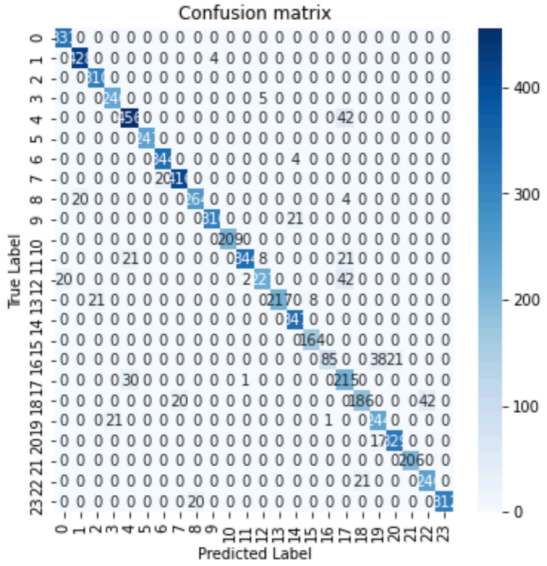
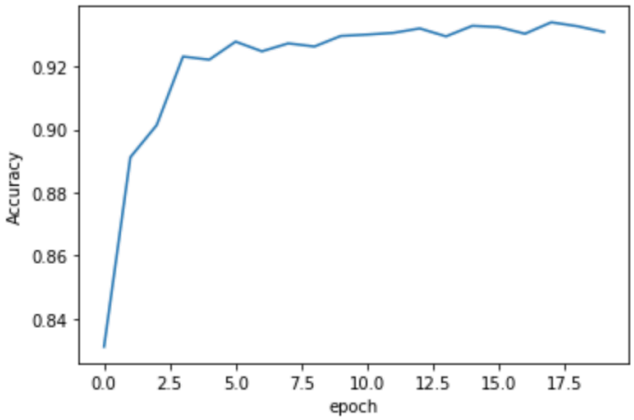


Figure 2: E, R and U Sign Language

The number of 20 epochs is applied and Table 2 shows how the accuracies moves when epoch increases. The accuracy increases as the number of epochs goes up but it eventually converges to 93% and it reaches 93% around 10 epochs. Therefore, it would be enough to have 10 epochs to train the images with the model which is implemented.

Table 2: Accuracy by epoch



3.2 Scores

The values of precision, recall, and F scores are compared, and all scores are calculated by labels as well.

Table 3: Scores

	precision	recall	f1-score	support
A	0.94	1.00	0.97	331
B	0.96	0.99	0.97	432
C	0.94	1.00	0.97	310
D	0.92	0.98	0.95	245
E	0.90	0.92	0.91	498
F	1.00	1.00	1.00	247
G	0.95	0.99	0.97	348
H	0.95	0.95	0.95	436
I	0.93	0.92	0.92	288
K	0.99	0.94	0.96	331
L	1.00	1.00	1.00	209
M	0.99	0.87	0.93	394
N	0.95	0.78	0.85	291
O	1.00	0.88	0.94	246
P	0.93	1.00	0.97	347
Q	0.95	1.00	0.98	164
R	0.99	0.59	0.74	144
S	0.66	0.87	0.75	246
T	0.90	0.75	0.82	248
U	0.82	0.92	0.86	266
V	0.94	0.95	0.95	346
W	1.00	1.00	1.00	206
X	0.85	0.92	0.89	267
Y	1.00	0.94	0.97	332
accuracy			0.93	7172
macro avg	0.94	0.92	0.93	7172
weighted avg	0.94	0.93	0.93	7172

As you see the scores on Table 3, precision reaches to 0.94 and f1-score reaches to 0.93. Moreover, most labels' precisions converge to 0.99 but some labels have very low numbers of precisions. For example, S has very low precision, 0.66 as we discussed with the confusion matrix, Table 1, because the sign was very confusing with other signs that would be one of the reasons why the precision is very low.

4 CONCLUSIONS AND FUTURE WORKS

With the model using Pytorch, detecting the American Sign Language (ASL) performs well. It takes a reasonable time to finish its task and the accuracy is pretty high with 94%.

Of course, ASL is the language using human's hands, therefore, the shapes of signs cannot avoid the limitations of usages of hands. Because of the reason, many of signs look similar with each other and it makes the machine confused. The CNN used above has very low precision when it is detecting letter 'S', so it would be great to find the reason why it is difficult to detect and improve the network.

With the network, which could detect ASL, people can invent a huge number of applications. For example, a translator can be invented as well. People can directly translate sign languages so they do not have to learn sign languages. Moreover, with this model, it would be great to extend the kinds of sign languages to other languages.

REFERENCES

[1] Sign Language MNIST. 2017. Kaggle. https://www.kaggle.com/datamunge/sign-language-mnist?select=sign_mnist_test

[2] Snapshot of Deaf and Hard of Hearing People, Postsecondary Attendance and Unemployment. 2011. Gallaudet University. https://www.gallaudet.edu/office-of-international-affairs/demographics/deaf-employment-reports#:~:text=Across%20all%20age%20groups%2C%20approximately,over%2065%20years%20of%20age.

[3] Statistics of The Deaf and Dumb. 1884. The New York Times, page 12. https://www.nytimes.com/1884/10/19/archives/statistics-of-the-deaf-and-dumb-.html

[4] The Best Apps for Learning Sign Languages. 2019. Healthy Hearing. https://www.healthyhearing.com/report/47829-The-best-apps-for-learning-sign-language

[5] Finetuning Torchvision Models. PyTorch. https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html.

[6] Writing Custom Datasets, Dataloders And Transforms. PyTorch. https://pytorch.org/tutorials/beginner/data_loading_tutorial.html

[7] Pytorch-Tutorial. (2018) GitHub. https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/02-intermediate/convolutional_neural_network/main.py