Using Machine Learning Methods to Recognize American Sign Language

Tianda Fu
Department of Electrical and Computer Engineering
Queen's University
Kingston, Ontario Canada
21tf5@queensu.ca
ELEC 872
2021/12/8

***Abstract*—American Sign Language (ASL) is the most commonly used sign language in the United States, the English-speaking regions of Canada, and parts of Mexico. Like other sign languages, the grammar and grammar of American Sign Language are different from the local spoken language. There is currently no reliable survey confirming the native speakers of American Sign Language; it is generally estimated to be between 200,000 and 2 million.**

So, what I want to do is to use machine learning methods such as Convolutional Neural Networks to identify the meaning of each gesture automatically.

Keywords—interactive system, machine learning, CNN, VGG-16, ASL, classification, image identification

I. Introduction

American Sign Language (ASL) is the most commonly used sign language in the United States, the English-speaking regions of Canada, and parts of Mexico. Like other sign languages, the grammar and grammar of American Sign Language are different from the local spoken language. There is currently no reliable survey confirming the native speakers of American Sign Language; it is generally estimated to be between 200,000 and 2 million.

Same as most countries in the world, deaf children in hearing families in the United States often use spontaneous family gestures for simple communication; but today, many middle schools and universities in the United States offer American Sign Language courses. American Sign Language is a language different from English-it has its own grammar and grammar, and has its own unique culture. The origin of modern American Sign Language depends on various historical factors and events, including education for the deaf, the hand gestures used by the Natives of North America, the unique situation of a small island in Massachusetts, and a priest trying to teach a deaf girl, Of course, the creativity and wisdom of deaf people who use sign language themselves are indispensable.

Although gestures seem to express meaning clearly, the concept of gesture expression can be as abstract as spoken language. For example, hearing children sometimes use the word "you" to express themselves, because others call them "you"; children who have learned the "you" gesture (pointing at each other with the index finger) will make the same mistakes that pointing to others to express yourself. And this shows that in American Sign Language, pointing gestures can be as abstract as spoken vocabulary.

In this project, the most important thing is to do some research on "Fingerspelling". ASL possesses a set of 26 signs known as the American manual alphabet. And this sign language alphabet can be used to spell out words from the English language[1]. Such signs make use of the 19 handshapes of ASL. For example, the signs for 'p' and 'k' use the same handshape but different orientations. And some of these kinds of features make a lot of deaf people confused, especially those deaf kids.

In this background, we tried to train a machine learning model to identify the meaning of each input gesture in the ASL alphabet. By using a convolutional neural network and a set of image data of all kinds of letters in the ASL alphabet.

II. Previous work

One important is that the most natural way to express emotion and information is through body movements such as signs and gestures [2][3]. Although there is a lot of information which can express by speaking and cannot convey by simple signs or gestures, this is still a very efficient way to use among those deaf and hearing-impact people[4].

In a lot of previous work, the accuracy of identification was not good enough by using only images. So, in the last decade, there are mainly two different ways to enhance the recognition accuracy of the American sign language alphabet. The first solution is adding information of depth.

The latest development of various sensor types, especially those that rely on depth information, has led to the development of many real-time applications, such as gesture and sign language recognition[5][6]. Due to the low cost, sensors such as Microsoft Kinect and Jump Motion Controller are widely spread and used by many researchers[7][8][9][10]. Sign language recognition problems can be divided into three sub-problems: sentences, isolated words and alphabet letter recognition[11]. In this paper, we only focus on the recognition of the American Sign Language (ASL) alphabet.

Tao et al.[12] proposed another way to strengthen the features extracted only from pictures. They first train a model to generate a set of signs in different viewing angles from images with depth information. And by using those multi-view augmentations, they come up with a convolutional neural network which can recognize the meaning of signs in the American Sign Language alphabet. And this method shows good efficiency and also comes with state-of-arts accuracy.

However, the methods we mentioned are using different ways to strengthen the features they can extract from the data by adding the information of depth or adding a lot more different viewing angles of one image. The commonality is that all of them need information of depth, which might not always work. Cause despite the devices with sensors to capture depth information such as Microsoft Kinect and Jump Motion Controller seems widely used, we can still only get a pure image without any extra information most of the time. In this project, we build up a convolutional neural network for American Sign Language recognition. And this network can use and only use 2D images for the classification. In this way, we can save a lot of computing resources for preprocessing the dataset and can be much more suitable for all most all situations.

III. Model

Convolutional Neural Networks are a type of Feedforward Neural Network that includes convolution calculations with a multi-layer structure. It is one of the representative algorithms of deep learning[13]. The convolutional neural network has the ability to represent learning and can perform shift-invariant classification of input information according to its hierarchical structure, so it is also called a "shift-invariant artificial neural network (SIANN)".

The research on convolutional neural networks began in the 1980s and 1990s. Time delay networks and LeNet-5 were the earliest convolutional neural networks; after the 21st century, with the introduction of deep learning theory With the improvement of numerical computing equipment, convolutional neural networks have been developed rapidly, and have been used in computer vision, natural language processing and other fields[14].

The convolutional neural network mimics the biological visual perception mechanism construction, which can perform supervised learning and unsupervised learning.

In this project, we are focusing on using supervised learning for image recognition in the area of computer vision. And normally a completed convolutional neural network gets an input layer, convolution layer, nonlinearity, pooling layer, fully connected and soft-max layer.

A. *Proposed model*

To identify the meaning of every sign in the American Sign Language alphabet, we built a CNN model in Fig. 1.

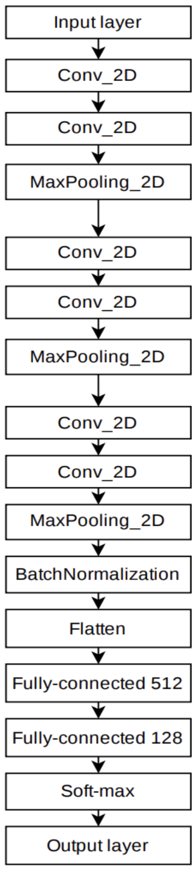


Fig. 1. The overall structure of the proposed model

1) *Feature extraction*

This model contains the input layer to read the input data and then put them into the convolutional neural network. And after that is the part of the feature extraction. In this part, we put three very similar sections and each of them consists of two two-dimensional convolution layers which use “ReLU”[15][14] as their activation function as well as a two-dimensional max-pooling layer. The main difference between these three sections is the number of the convolution kernel. The first two two-dimensional convolution layers use 16 and 32 convolution kernels respectively, the second two two-dimensional convolution layers use 32 and 64 convolution kernels respectively and the last two two-dimensional convolution layers use 128 and 256 convolution kernels respectively. By doing this, we can extract as many global features as we want. Between each two, there is a max-pooling layer with a pooling size of (3, 3), which can reduce the size of data by 1/9 and boost the efficiency of feature extraction.

2) *Normalization*

The normalization of data is a common step in the preprocessing of neural network input pipelines, but in deep networks, as the input data is passed step by step in the hidden layer, its mean and standard deviation will change, resulting in covariate shift, which is considered to be one of the reasons for vanishing gradients in deep networks[16]. Batch Normalization partially solves such problems at the cost of introducing additional learning parameters. In my model, there is a Batch Normalization layer after the step of feature extraction to avoid vanishing gradients.

3) Classification

In the part of the classification, the model first flattened the data output from the Batch Normalization layer, and then put them into a fully-connected layer with 512 neurons. The next layer is also a fully-connected layer with 128 neurons. Between these two fully-connected layers, this model would randomly drop out 20% of all the neurons to avoid overfitting. And after that, we can use a softmax function to do the classification. Actually, softmax can map the output of multiple neurons to the (0,1) interval, which can be understood as a probability to perform multi-classification.

IV. Experiments

My primary goal is to identify the represented letter of a sign in the American Sign Language alphabet by and only by several images. As my target, we use an opensource dataset called ASL Alphabet Synthetic

A. ASL Alphabet Synthetic Dataset

The dataset is divided into directories accordingly to each alphabet letter. Each directory contains 3 sub-directories:

*1) Lit: 1k .png images 640*480 representing realistic renders*

2) Segmentation: 1k .png images representing the segmentation of parts of interest (fingers and hand)

3) Annotation: 1k .json files representing scene metadata.

Those images are rendered from randomized 3D hand models which got lots of randomized parameters such as texture, roughness, color, nails and clothing. Also, those images got randomized backgrounds and illumination. Besides that, the camera position in relation to the hand has a relative random transform.

B. Preprocessing

In this project, we have only done three kinds of preprocessing. The first one is resizing the image into sizes of (128, 128, 3) after reading those images into memory. And that would compress the total pixel number down to nearly 5% of the original data. Besides that, the values of each channel are been compressed from (0, 255) to (0, 1). Another preprocessing is targeting labels. To use categorical cross entropy as the loss function[17], we use the OneHot encoder transforming the original labels such as “a”, and “b” into numbers like “1” and “2”.

C. Experiments

For this project, only a part of the ASL Alphabet Synthetic dataset has been used. And none of the data in the directories of segmentation and annotation are used.

1) How does the step of normalization affects the result

Batch Normalization, very similar to ordinary data normalization, is a method of unifying scattered data as well as a method of optimizing neural networks. And this kind of data with uniform specifications can allow machine learning to learn the laws in the data much easier.

So, in the first several experiments we didn’t use Batch Normalization as an extra layer after the feature extraction step. And that made my experiment result with an unusable model which got only 3.861% of accuracy.

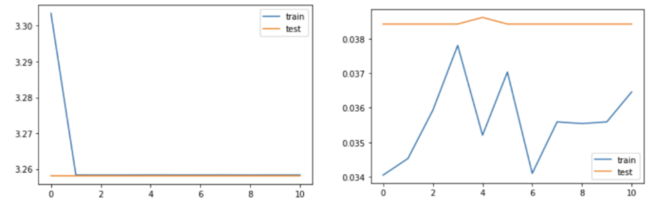


Fig. 2. The loss and the accuracy result of the model without BN

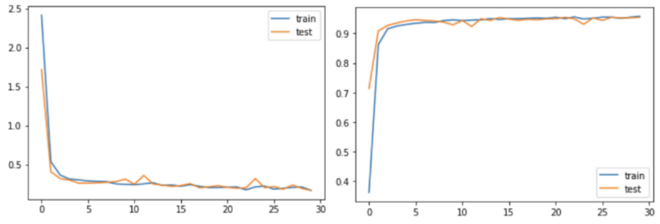


Fig. 3. The loss and the accuracy result of the model with BN

However, after we add a Batch Normalization into the model, it turns out to be a very good result the accuracy of the model can reach over 96%.

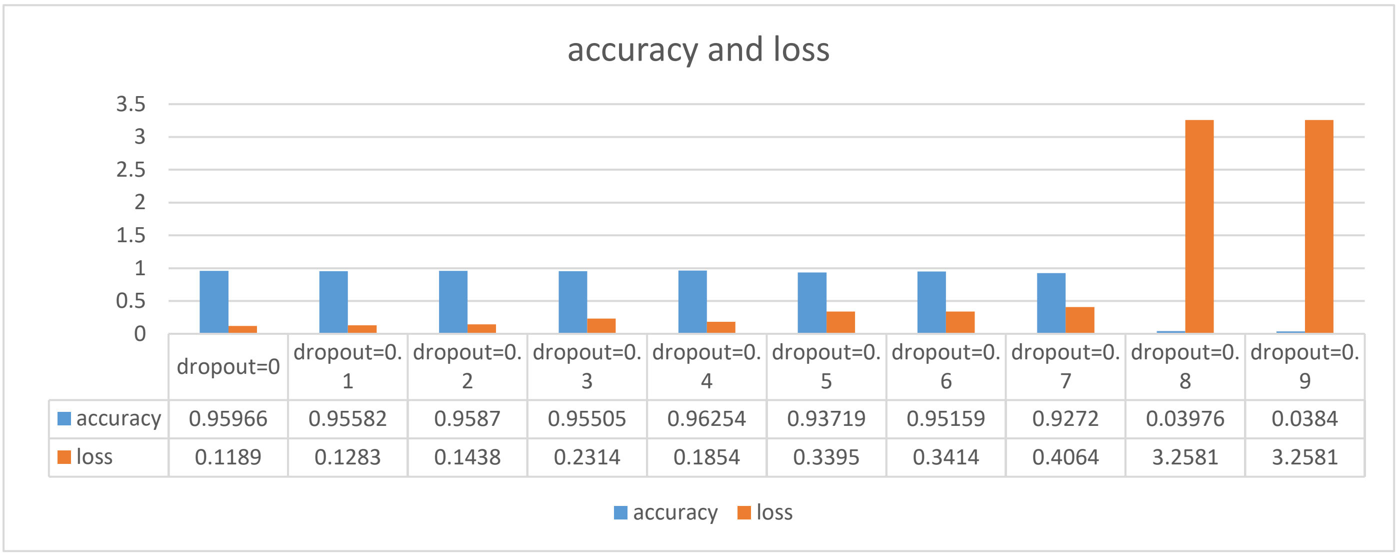
Comparing the results shown in fig. 2 and fig. 3, we can find that no matter for the train data or test data, the result tends to converge after 3 to 4 epochs in fig. 3. But the results in fig. 2 looks almost like a straight line. And that might be caused by vanishing gradients, which can be perfectly solved by using a normalization function.

2) How does the dropout affect the result

In a machine learning model, if the model has too many parameters and too few training samples, the trained model is prone to overfitting. The problem of overfitting is often encountered when training neural networks. Overfitting is specifically manifested in: the model has a smaller loss function on the training data and higher prediction accuracy; but on the test data, the loss function is relatively large, and the prediction accuracy is low.

Overfitting is a common problem in many machine learning. If the model is overfitted, then the resulting model is almost unusable. In order to solve the problem of overfitting, the method of model integration is generally used, that is, training multiple models to combine. At this time, the time-consuming training model becomes a big problem. Not only is it time-consuming to train multiple models, but it is also time-consuming to test multiple models.

However, dropout can effectively alleviate the occurrence of overfitting, and achieve the effect of regularization to a certain extent. So in this part of the experiment, we discuss the relationship between the dropout values and the results.



	dropout=0	dropout=0.1	dropout=0.2	dropout=0.3	dropout=0.4	dropout=0.5	dropout=0.6	dropout=0.7	dropout=0.8	dropout=0.9
🟦 accuracy	0.95966	0.95582	0.9587	0.95505	0.96254	0.93719	0.95159	0.9272	0.03976	0.0384
🟧 loss	0.1189	0.1283	0.1438	0.2314	0.1854	0.3395	0.3414	0.4064	3.2581	3.2581

Fig. 4. The result when using different dropout values

As fig. 4 shows that the accuracy would meet a significant drop from almost 93% down to less than 4%. And that might be because when the dropout is bigger than 0.8, too much information is dropped and there will not be enough features for the model to train a decent result.

3) Compare with other models

In this section, we used several existing models which are mainly focused on image recognition to test their accuracy of them. After that, we can compare our model with them.

a) VGG-16

VGG was proposed by the Visual Geometry Group[18] of Oxford. This network is related to work on ILSVRC 2014. The main work is to prove that increasing the depth of the network can affect the final performance of the network to a certain extent. This network has two structures, namely VGG-16 and VGG-19. And there is no essential difference between the two, but the network depth is not the same.

An improvement of VGG-16 compared to AlexNet[19] is to use consecutive 3*3 convolution kernels to replace the large convolution kernels in AlexNet (11*11, 7*7, 5*5). For a given perceptual field (the local size of the input image related to the output), using a stacked small convolution kernel is better than using a large convolution kernel, because multiple non-linear layers can increase the depth of the network to ensure learning more complex models, and the cost is relatively small (fewer parameters).

To put it simply, in VGG, three 3*3 convolution kernels are used instead of 7*7 convolution kernels, and two 3*3 convolution kernels are used instead of 5*5 convolution kernels. The main purpose of this is to ensure that under the condition of the same perceptual field, the depth of the network is improved, and the effect of the neural network is improved to a certain extent.

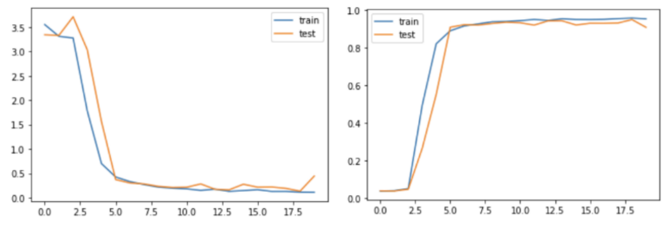


Fig. 5. The result of a VGG-16 model

In our model, we changed the input size from the original of (227, 227, 3) to (128, 128, 3) to fit our dataset. At the part of the output, we add a fully-connected layer with 512 neurons as well as another fully-connected layer with 26 neurons. By doing that, we can ensure the number of output classes is 26, which can fit our dataset.

Fig. 5 shows the result of using a customized VGG-16 model to classify the signs given in the American Sign Language alphabet. The highest accuracy of this model is about 95.5%. Which is a little bit lower than our proposed model. But another important difference between VGG-16 and our model is the time consumption. When batch size equals 64, VGG-16 needs 61 seconds for processing each epoch, while our model needs only 10 seconds for the same processing.

V. Analysis and Discussions

In this project, we used the dataset named ASL Alphabet Synthetic Dataset which has directories representing each letter, and each letter directory has three sub-directories one of them contains image data, another one contains segmentation data, the last one contains the sense's metadata. But during this whole project, we only used the image directory.

By ignoring the other two kinds of files, we can make the steps of preprocessing be much easier. But the drawback of this is that we didn't use the segmentation information, which also means we didn't use as many features as we could.

To use those segmentation data, we could build another neural network for semantic segmentation. Such as U-Net[20], we could use that convolutional network training a model specified for separate the hand or even fingers from the image background. After get a image segmentation step, we could put the separated segmentation image into a convolutional neural network. We believe that we can get a higher accuracy if we take the segmentation image as the input because the segmentation dataset can be much purer than the original one.

For now, this proposed model can recognize images without depth information, and that already has a much bigger applying field than those researchers which can only identify images with depth information. because in our opinion, in most cases of using American Sign Language for communication, they would use images without depth information.

VI. Conclusion

This project mainly focuses on identifying the meaning of signs in the American Sign Language alphabet. We proposed a convolutional neural network to train the machine learning model for classification. This network can extract features from and only from image data but without depth information.

As for the result, there are two advantages:

- this model can reach over 96% accuracy on ASL Alphabet Synthetic Dataset, and that accuracy is even higher than VGG-16, which is a famous network for image classification.
- Compare to VGG-16, this network has much fewer parameters. And that results in the different running times between these two neural networks. Taking 64 as the batch size, the VGG-16 model needs 61 seconds for training, while our model needs only 10 seconds. It also shows that we used a much easier model to reach the same goal and get an even better result.

Although we have a really good result, some parts of this model can still be improved. Despite we didn't meet the overfitting problem, we can't get an accuracy higher than 96.2% no matter how we change the dropout, optimizer, activation function or the number of neurons in fully-connected layers. However, we believe that we could improve the result by applying edge detection. Because after the edge detection and semantic segmentation, we could extract the hand part out of the test images and even recognize the position of each finger. By doing that, we can totally remove the bias contained in the image background, so after those steps, basically every single feature extracted by convolution layers can be useful.

References

[1] Costello, Elaine. Random House Webster's American Sign Language dictionary, 2008

[2] H. S. Badi and S. Hussein, "Hand posture and gesture recognition technology", Neural Comput. Appl., vol. 25, no. 4, pp. 871-878, 2014.

[3] M. J. Cheok, Z. Omar and M. H. Jaward, "A review of hand gesture and sign language recognition techniques", Int. J. Mach. Learn. Cybern., vol. 10, no. 1, pp. 131-153, Jan. 2017.

[4] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: A survey", Artif. Intell. Rev., vol. 43, no. 1, pp. 1-54, Jan. 2012.

[5] J. Han, L. Shao, D. Xu and J. Shotton, "Enhanced computer vision with microsoft Kinect sensor: A review", IEEE Trans. Cybern., vol. 43, no. 5, pp. 1318-1334, Oct. 2013.

[6] N. H. Dardas, and N. D. Georganas. Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques. Instrumentation and Measurement, Vol. 60, pp. 3592-3607, 2011.

[7] N. Pugeault, and R. Bowden. Spelling It Out: Real-Time ASL Fingerspelling Recognition. 2011 IEEE Workshop on Consumer Depth Cameras for Computer Vision, pp. 1114-1119, 2011.

[8] P. Gurjal, and K. Kunnur. Real Time Hand Gesture Recognition Using SIFT. International Journal of Electronics and Electrical Engineering, Vol. 2, Issue 3, 2012.

[9] L. Xia and J. K. Aggarwal, "Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 2834-2841, Jun. 2013.

[10] Z. Ren, J. Yuan, J. Meng, and Z. Zhang. Robust Part-Based Hand Gesture Recognition Using Kinect Sensor. IEEE Transactions on Multimedia, pp. 1110-1120, 2013.

[11] W. Aly, S. Aly and S. Almotairi, "User-Independent American Sign Language Alphabet Recognition Based on Depth Image and PCANet Features," in IEEE Access, vol. 7, pp. 123138-123150, 2019.

[12] W. Tao, M. C. Leu, Z. Yin, American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion, Engineering Applications of Artificial Intelligence, Vol. 76, pp. 202-213, 2018

[13] Gu et al., Recent advances in convolutional neural networks. arXiv preprint arXiv:1512.07108, 2015

[14] A. Ng, K. Kian, and B. Younes, Convolutional Neural Networks, Deep learning. Coursera and deeplearning.ai, 2018

[15] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Icml. 2010.

[16] Sergey Ioffe, Christian Szegedy, Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:448-456, 2015.

[17] Zhang, Zhilu, and Mert R. Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels." 32nd Conference on Neural Information Processing Systems (NeurIPS). 2018.

[18] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv: 1409. 1556, 2014

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, pp. 1097-1105, 2012

[20] Olaf Ronneberger, Philipp Fischer, Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCA, pp 234-241, 2015