EE 541 - Computational Introduction to Deep Learning

Computer Vision for Reading American Sign Language

Project Report

Yuqi Chen, Mengqi Du

Dec 11, 2022

1 Introduction

1.1 What is ASL

Communication with the deaf and hard of hearing community is quite a problem nowadays. ASL(American Sign Language) has been the most common and predominant way for those deaf people to communicate with others in the United States. However, many people don't know sign language and it may cost a great effort for them to learn. In order to solve this problem, we want to build a model to help recognize and interpret static hand gestures.

ASL divided hand postures to 28 different meanings (26 alphabet letters, del and space). So in our project, we constructed the model with 29 different classes(the 28 different meanings above and the meaning of nothing).

Figure 1: ASL finger alphabet

1.2 Related work

Many scholars have made many outstanding contributions to this field. For this part, we are going to introduce several related works that deal with similar problems to our project.

Gamal Tharwat, Abdelmoty M. Ahmed and Belgacem Bouallegu use the vision-based approach to recognize the hand gesture which consists of four stages : the stage of data processing, preprocessing of data, feature extraction, and classification[1]. They achieved an accuracy of 99.5% for the K-Nearest Neighbor (KNN) classifier.

M. M. Kamruzzaman constructed a system by applying CNN for the recognition of Arabic hand sign-based letters and translating them into Arabic speech[2]. The designed system will automatically detect hand sign letters and speak out the result with the Arabic language with a deep learning model. This system gives 90% accuracy to recognize the Arabic hand sign-based letters.

Bekalu Tadele Abeje, Ayodeji Olalekan Salau, Abraham Debasu Mengistu present the development of a brand new sign language recognition system which can translate Ethiopian sign language to Amharic alphabets using computer vision technology and CNN[3]. The system they constructed consists of three stages : preprocessing, feature extraction, and recognition.

2 Dataset

2.1 About Dataset

The dataset used in this project is retrieved from American Sign Language on Kaggle. It consists of a training data set and a test data set. The training set is a collection of 87,000 images separated in 29 folders, each of which represents a corresponding class. Among them, 26 classes are for the letters A-Z and 3 classes

for *SPACE*, *DELETE* and *NOTHING*. The test set is a collection of 28 images for model evaluation. 29 different sign language gestures are shown in Figure 2.

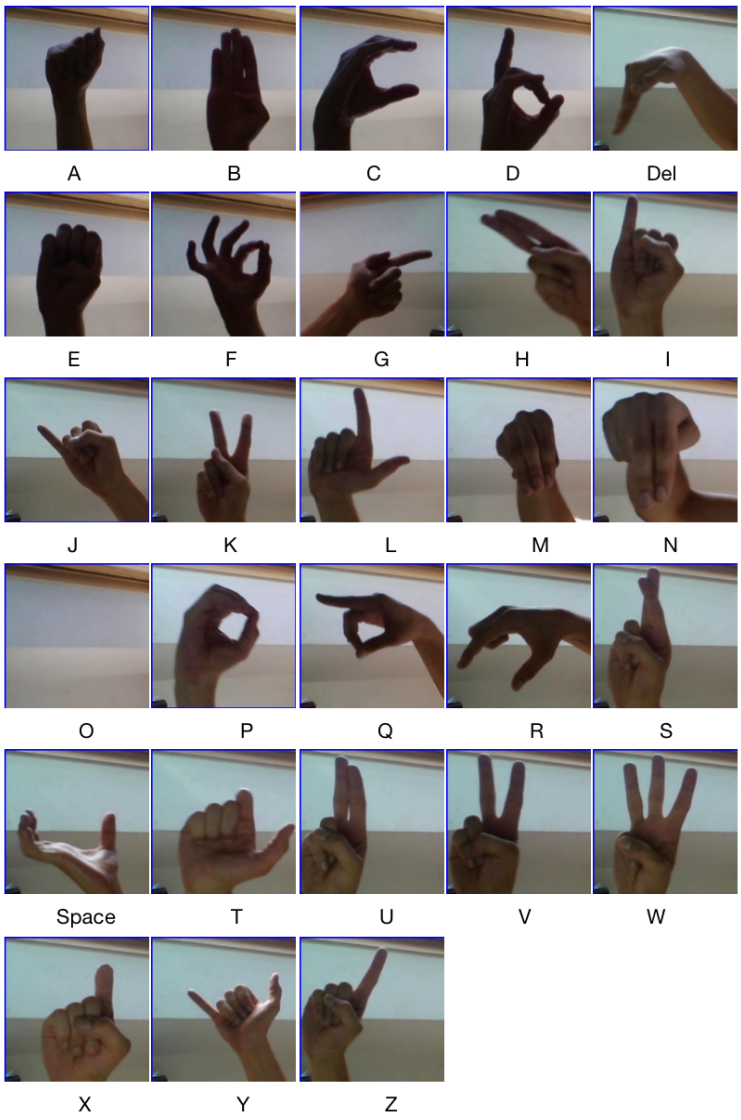

Figure 2: ASL finger alphabet in dataset

2.2 Data Preprocessing

For computational simplicity, image scaling and image grayscale are used to change the image complexity from 200 × 200 pixels, 3 channels to 64 × 64 pixels, single channel.

Image Scaling

Image resizing is the process of resizing the width and height of an image. OpenCV provides an interpolation method of resampling using pixel area relation to help convert a 200 × 200 pixels to a 64 × 64 pixels image. This interpolation method divides each image into separate regions and the value of each region is replaced with the average value of this region.

Image Grayscale

Grayscaling is the process of converting an image from other color spaces to shades of gray. OpenCV also provides a method to convert original images of RGB to GrayScale.

Obviously, both scaling down the pixel number and reducing channels will result in a visible quality loss. However, considering the tradeoff between the computational complexity and recognition performance, the size of images can be decreased in a way that the accuracy is still maintained fairly well.

3 Model Structure

3.1 AlexNet Module

AlexNet is a convolutional neural network architecture. It contains eight layers. The first five layers are convolutional layers, with the first, second and fifth layer followed by a max-pooling layer. And the last three layers are fully-connected layers, which utilize the non-saturating ReLU activation function. The basic structure of our implementation is shown in Figure 3.

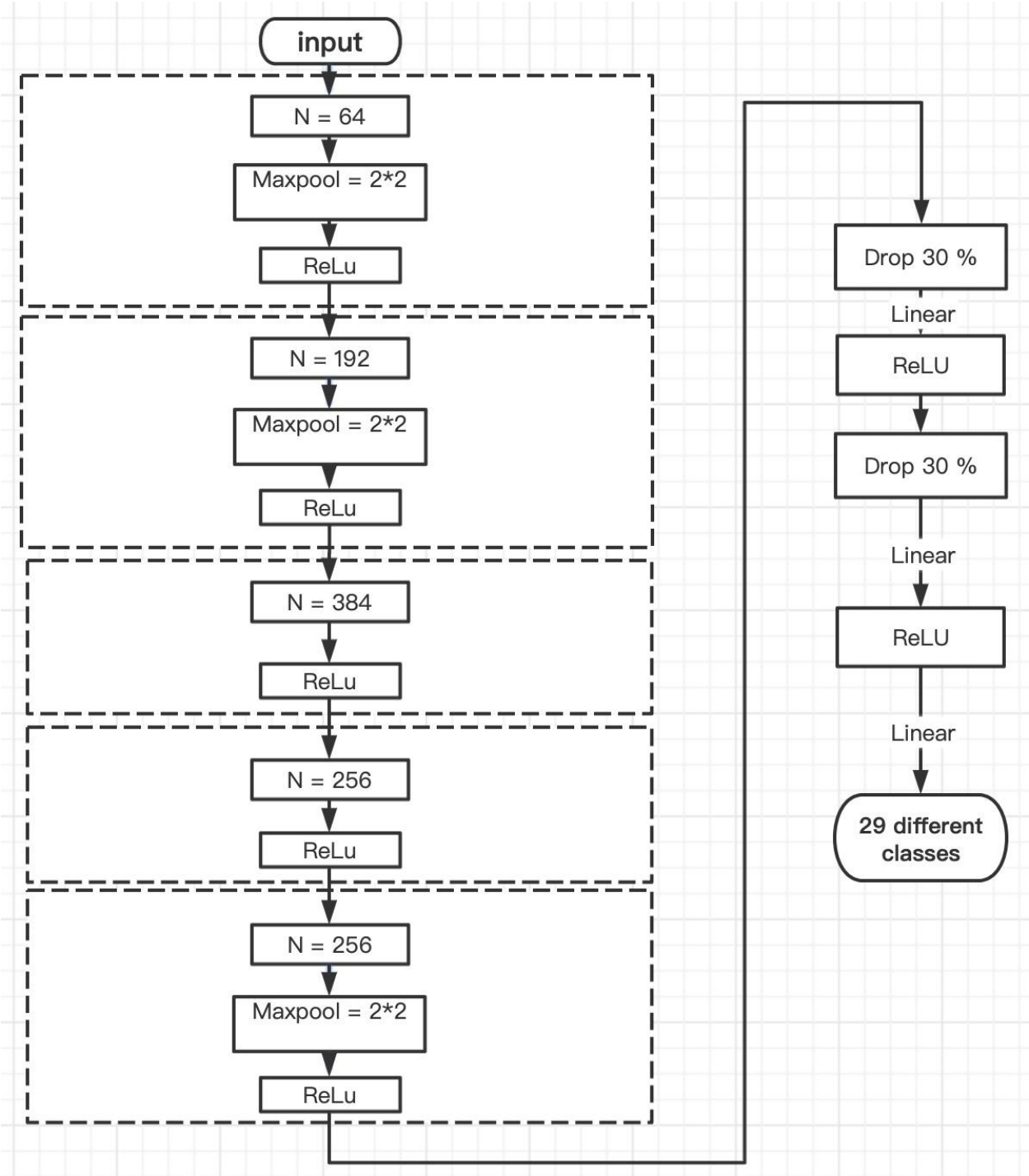

Figure 3: Model structure of AlexNet

3.2 ResNet v1.5 Module

ResNet v1.5 is a modified version of the original ResNet v1 model, which presents a residual learning framework to ease the training of substantially deep networks. The key difference is that v1.5 has stride = 1.5 in the 3×3 convolution in the bottleneck blocks, which makes it slightly more accurate but incurs a small performance drawback. The basic structure is shown in Table 1.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1: Architecture of ResNet 50 v1.5

4 Implementation and Results

4.1 Hyperparameters

Our project mainly focuses on the implementation and evaluation of the AlexNet Module and we'll take the ResNet v1.5 Module built in Pytorch as our baseline of analysis. So we experimented with different hyper-parameters to finetune the AlexNet module to see how these parameters take effect on the outcome. And finally we'll compare these two modules in terms of overall performance.

Learning rate

We have experimented with different values of the learning rate on the AlexNet model. The accuracy plot under different learning rates is shown in Figure 4.

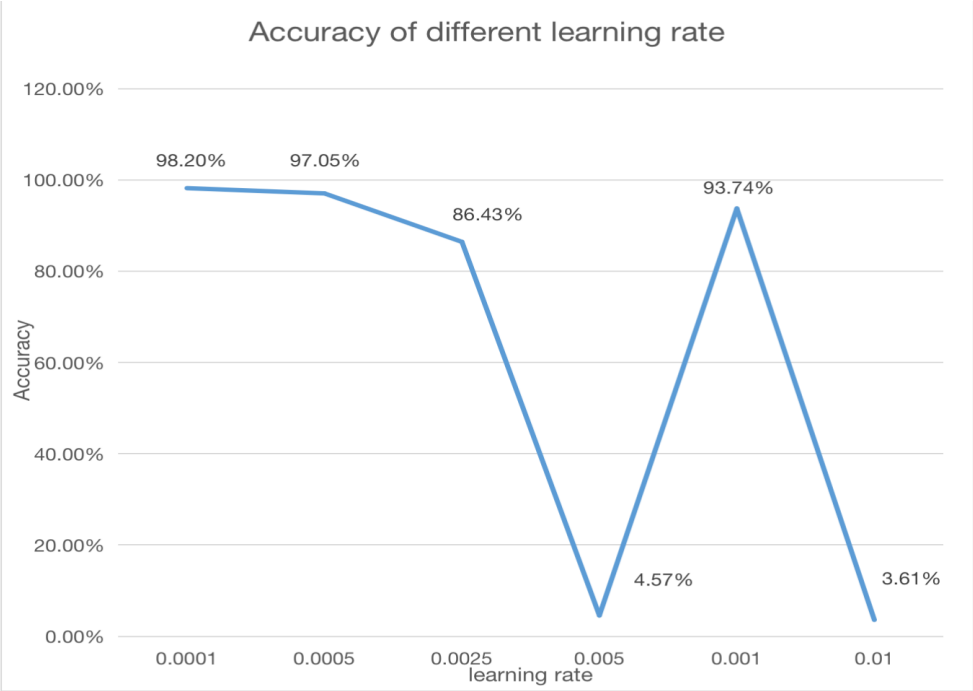

Figure 4: Learning rate-Accuracy plot

From the result above we can find out that the learning rate 0.0001 gave us the highest accuracy. So we finally chose 0.0001 as our learning rate.

Stride

The stride in a CNN defines the step size of the kernel when traversing the image. We experimented with different values of the first layer stride. The accuracy table is shown in Table 2.

Stride	1	2	3
Accuracy	98.20%	97.10%	N/A

Table 2: Comparison of different stride

From the result we can conclude that the stride influences the convergence of the training model. And the accuracy will decrease along with the increase of stride.

4.3 Overall comparison

For this part, we'll compare our fine-tuned AlexNet Model with RestNet Module. The learning rate we used for both models is 0.0001. We recorded the accuracy every other ten batches(batch = 100). And the accuracy plot is shown in Figure 5 and Figure 6 respectively.

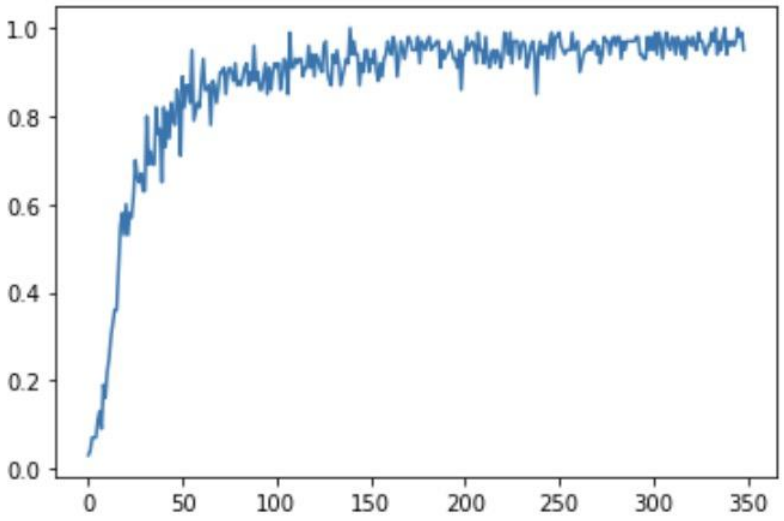

Figure 5: Accuracy in the training process for AlexNet

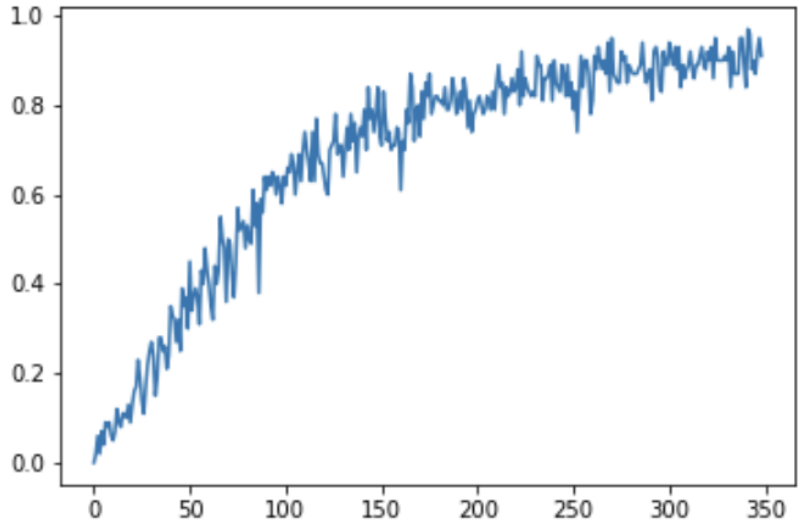

Figure 6: Accuracy in the training process for RestNet

Above all, after training the dataset used in our project, there's only a minor difference regarding the final performance. Both modules perform fairly well on the ASL recognition task while the AlexNet module has a slightly higher accuracy over the RestNet module. Also, AlexNet reached convergence faster than RestNet.

5 Conclusion

From this project, we have learned a lot about data preprocessing, CNN hyper-parameters and many different learning strategies. Because the training data is limited, our model seems to have good performance by testing them, it may not work well in the real world. For the data preprocessing part, we found out that this part would not lead to much information loss. Just like the ASL classification part, we resize the pictures, but it doesn't seem to have a significant effect on the result. For the hyper-parameters part, we should try and modify our parameters many times to get the preferred result. And different models may be suitable for different tasks, so when we are faced with different situations, we need to choose the right model which may have the best performance.

6 References

[1] Li, Yang and Tharwat, Gamal and Ahmed, Abdelmoty M. and Bouallegue, Belgacem. Arabic Sign Language Recognition System for Alphabets Using Machine Learning Techniques. In 2021 Journal of Electrical and Computer Engineering.
[2] Kamruzzaman, M. M. Arabic Sign Language Recognition and Generating Arabic Speech Using Convolutional Neural Network . In 2020 Wireless Communications and Mobile Computing
[3]Bekalu Tadele Abeje, Ayodeji Olalekan Salau, Abraham Debasu Mengistu.In 20 Aug 2022, Multimedia Tools and ApplicationsVolume 81 Issue, pp 29027–29043.
[4]https://github.com/NVIDIA/DeepLearningExamples/blob/master/PyTorch/Classification/ConvNets/resnet50v1.5/README.md