CS274C Project–Hand Gesture Recognition

Xiaoran Li, Xudong Li, Yao Zhang, Xiaoxuan Zhang, Bowen Sun
Department of Electrical Engineering and Computer Science @ UCI

1 Introduction

Our group is working on hand gesture recognition. The goal is when we input a hand gesture image, the program can recognize what kind of hand gesture the input is using well-trained neural network.

The project can be used in image recognition for the deaf-mutes. Besides, it can also be applied to the battlefield, like letting robots imitating actions of human beings and tracking the actions of enemies. In these situations, the input datasets will be large-scale, so we grab MapReduce[1, 2] structure to solve the main problem. The aim of using MapReduce is to save the communication cost which save the accuracy and increase the speed for the whole net.

2 Data

2.1 Dataset

To make the recognition better, we want to find the dataset with the feature angel, so we first search the dataset on the website. However, this kind of dataset is too hard to find, so we choose to build the dataset by ourselves. We write the code which can capture the image of our hand gestures which consists of rocks, scissors, papers with different angles. Our databset consists of 2700 images.

Figure 1: Hand gesture dataset

2.2 Preprocess

Preprocess consists of three parts: Graying, Gaussian blur and adaptive threshold.

When dealing with images, we pay more attention to the image's information of edge gradient instead of information of color. After graying, the matrix dimension decreases, which improves the speed of operation.

Gaussian blur is a process we get weighted average value of pixel point data of images. Every new pixel point is a weighted average of every neighbor pixel point surrounding it. The goal of Gaussian blur is reduce

the influence of Gaussian white noise.

People usually use a global value as threshold value. But it may not be good in all the conditions where image has different lighting conditions in different areas. In that case, we apply adaptive thresholding. According to a small regions of the image, we calculate the thresholding. So we get different thresholds for different regions of the same image and it gives us better results for images with varying illumination.

3 Architecture Design and Training

Assuming multi-user is trying to load the program through the website. From user side, they are expecting when user creates a input, the output will generate immediately. Thus, using distributed file system to do computing and calculation will be a fast and simple way to solve the problem.

We use MapReduce to fast compute and communicate between different servers. In our case we create three servers. Each server runs by themselves for training and validation. After the value trained and transferred to the desire server(shuffle phase), each server reduces the transferred value to find the original data value. Taking an input value to the program then using the idea of reverse index find the original matched place.

Under each server, we arrange a certain neural network with same model and then distribute the training set to each server. The method to allocate mission here is mapreduce algorithm. After that, each server could train and fix its own model parameter with its unique data sets.

We designed a 11-layer NN for each server to load and run, this 11-layer NN is shown as Figure 2. The input of this NN is $300 \times 300 \times 1$ gray image (after preprocessing), it has four convolution layers(kernel size: 3×3, 5×5), each convolution layer is followed with a max-pooling layer that chose the largest number in a 2×2 input array as the output to next layer, after which there are three fully-connected layers with a 12-class softmax function as the final output. For activation function, we adopt ReLU[3] to ensure the existence of gradient. We also add dropout[4] layers with parameter of 0.5 before each dense layer to handle overfitting problem, as well as the training efficiency.

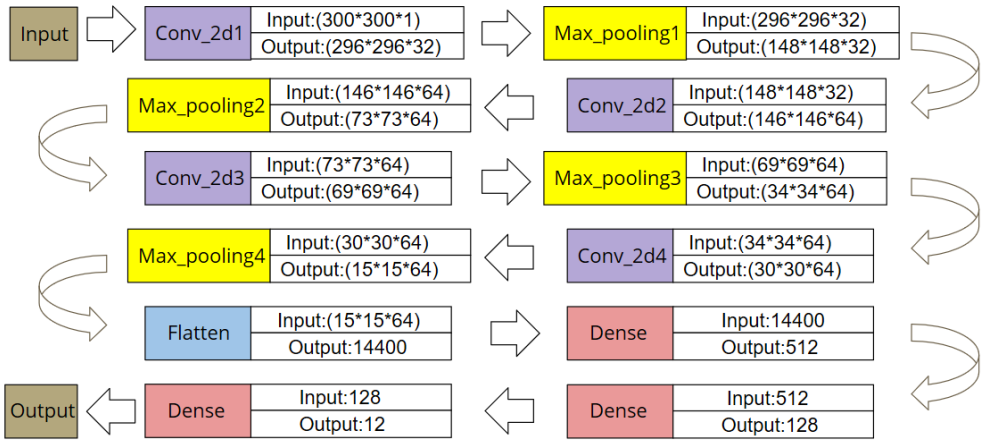

Figure 2: CNN architecture

4 Results

We use 90 percent of images in our dataset for training and 10 percent of images in our dataset for testing. Curves of loss functions and accuracy are shown below.We can see clearly from Figure 3, after

10 times epochs, the accuracy of training can achieve 97 percent. The accuracy is converge to 1. Using OpenCV, we implemented real-time recognition of gestures as Figure 4.

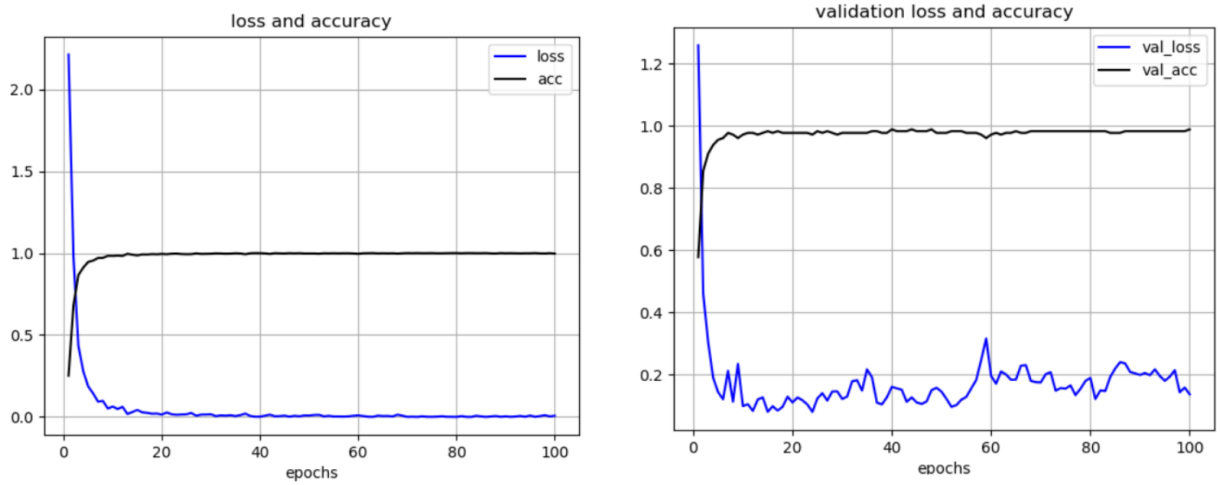

Figure 3: Loss and accuracy (left: training result, right: testing result)

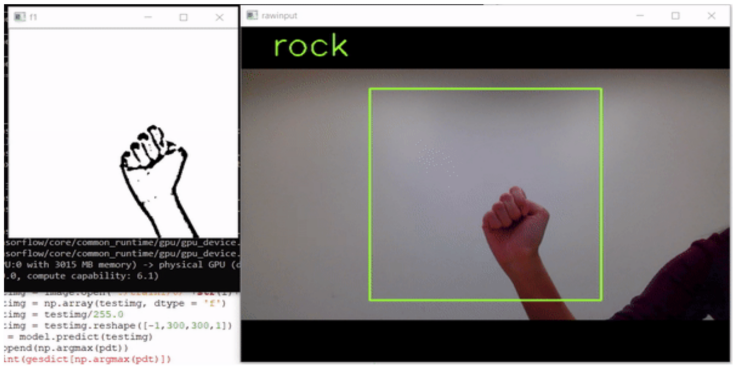

Figure 4: Real-time hand gesture recognition

5 Conclusion and Discussion

MapReduce model requires large set of data for acquisition. Each server runs small part of data. After combination of these parts, we get the target model. Meanwhile using the trick name to reverse index will increase the complicity of the program.

For future work, we can use reverse index to find from a hand gesture to the target resource in our database. And we can optimize our original CNN architecture like changing the number of pooling layers and convolution layers, using architecture like ResNet[5] or changing our activation function into LeakyReLU[6] function for improving the accuracy of recognition since it is possible that some neurons can never be activated with ReLU. Besides, we could focus our attention on trying to recognize of dynamic action instead of images, which can be used in multiple fields like medical and education.

References

[1] Songze Li, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr. Coded MapReduce. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2015*, 2016.

[2] Gleb Skobeltsyn Claudio Lucchese and Wai Gen Yee. The 7th workshop on large-scale distributed systems for information retrieval (LSDS-IR'09). *ACM SIGIR Forum*, 2009.

[3] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. *JMLR W&CP*, 2011.

[4] Ilya Sutskever, Geoffrey Hinton, Alex Krizhevsky, and Ruslan R Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] Andrew L. Maas, Awni Y. Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML '13*, 2013.

Contribution

Xiaoran Li built the algorithm for the network in mapreduce. Xudong Li collected the database and preprocessed the database. Yao Zhang designed the network architecture. Xiaoxuan Zhou trained our dataset. Bowen Sun analyzed the results.