An Implementation of ResNet on the Classification of RGB-D Images

Tongyan Gong[1,⋆],Huiqian Niu[2]

[1] Guizhou University of Finance and Economics
2278802241@qq.com
[2] Technology and Data Middle Platform, JD.com
niuhuiqian@jd.com

Abstract. Facial recognition is to identify human faces from an image. It is becoming more and more important these days as it can be applied in multiple industries, such as bank, airport, e-business, etc. Because of the broad application prospects, face recognition is actively developed and researched by many people, companies and academic organizations. In this paper, we customize one of the facial recognition models developed in the recent years – ResNet on the Intellifusion 3D face dataset[1]. And then we evaluate the performance of the algorithm by adjusting the depth of the network, pre-processing steps of the pictures and also the learning rate.

Keywords: ResNet· Computer Vision· RGB-D· Face Recognition.

1 Introduction

There has been a long research history in the area of computer vision and facial recoginition. In 1987, Sirovich and Kirby used Eigenface [2] in facial recognition for the first time. In 1998, Gary applied Continuously Adaptive Mean Shift (CAMSHIFT) algorithm for face tracking [5]. In recent years, the improvement of hardware resulted in a trend of using neural network for facial recognition. Krizhevsky has come up with convolution neural network to solve the image classifiication problem [6]. In 2015, Google published FaceNet [9], a new facial recognition model, which achieves record high accurary on Labeled Faces in the Wild (LFW) dataset. Kaiming developed a new type of deep neural network with residual block in the same year, which provides a way to resolve some issues brought by the depth of the network [7]. Besides the development of new models, there are also researches focusing on bring the facial recognization to more hardware platforms and scenarios. CMU released OpenFace face recognition library in 2016 [4] providing a new benchmark for mobile platform. Anil discusses the state-of-art and challenges of facial recognition technology in the area of crime investigations [8]. In this paper, we implement a CNN network based on ResNet for face recognition. And then we employ the Intellifusion 3D dataset to evaluate the

⋆ Tongyan Gong is the corresponding author.

performance of our proposed network getting 82%. The influence of learning rate and inference time under different devices (CPU and GPU) are explored in experiments. Our code is implemented and optimized on the AIBench and its hardware platform[24, 25, 27, 26, 28] and this article takes part in the AI System and Algorithm Challenge organized by Benchcouncil. The source code of AIBench is publicly available from `http://www.benchcouncil.org/benchhub/AIBench/` (Sign up to get access). BenchCouncil AI Challenge includes four tracks: International AI System Challenge based on RISC-V Subject[34], International AI System Challenge based on Cambricon Chip Subject[33, 32], International AI System Challenge based on X86 Platform Subject[29, 30, 21] , and International 3D Face Recognition Algorithm Challenge Subject[31].

2 Related Work

2.1 Convolutional Neural Network

Convolutional neural network (CNN) is commonly used in extracting patterns from images as images can usually be represented in 2D matrix and convolutional neural network can easily applied on data in the form of 2D matrix. The basic layers used in forwarding the convolutional neural network are the convolution layer, down-sampling layer and non-linear activation layer. [12] also explained how one can implement convolutional layer using fast Fourier transform and it adopted the cuFFT library. As for other embeded systems, optimized FFT libraries[13–15] can be used for CPUs, especially for inference. Convolution layer basically convolves the input feature maps (assume it as a 2D matrix) with learnable kernels. For a full convolution, the output of the layer is given by:

$$z(u,v) = \sum_{i=-\infty}^{\infty} \sum_{-\infty}^{\infty} x_{i,j} k_{u-i,v-j} \tag{1}$$

Kernel can be considered as a feature pattern that we want to see to what extend each part of image matches. After convolution layer, feature mapping is created. However, the data amount is still large as the number of original images in the training set is usually large . Down-sampling provides a way to decrease the size of the data while still keeping the invariance extracted by the convolution layer as much as possible. More formally:

$$x_j^l = f(\beta_j^l down(x_j^{l-1}) + b_j^l) \tag{2}$$

Concretely, there are usually two ways of down sampling: max pooling and average pooling. For max pooling, the down function is picking the maximum value of the input matrix as the output; for average pooling, the down function is calculating the average value of the input matrix as the output. The down-sampling layer also helps improve the model performance by reducing the potential of over-fitting. And then there is non-linear activation layer. One of the most common non-linear activation function is the ReLu function:

$$f(x) = max(0, x) \tag{3}$$

Finally, the output of the layer will abandon the negative values in the input. The superiority of the convolutional neural network has been demonstrated to be effective in the computer vision task, thus we use it in this paper to do face recognition.

2.2 ResNet

With the three basic layers mentioned in the last section, scientists and engineers can use them multiple times in any combination to their own neural network fitting into the problem to solve. However, deeper network generates new issues, overfitting, taking more time to train and the degradation problem: accuracy of the network decreases rapidly in deeper network. To solve this problem, Kaiming added residual blocks into the deep neural network [7].

As a result, the stacked nonlinear layers are trained to fit into the mapping of $F(x) := H(x) - x$ in stead of $F(x) := H(x)$. In other words, the original mapping is changed to F(x)+x from $F(x)$. This change is based on the assumption that it is easier to optimize the residual mapping with the identity mapping, serving as a "shortcut" mechanism [7].

3 Work Description

3.1 Design

Since the ResNet has deeper network structure compared with early work and has better performance, we use a ResNet-like network in the paper to improve performance. Figure 1 shows the workflow of the system developed for this work.

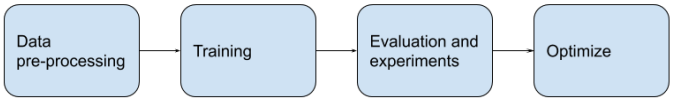

Fig. 1: design of the workflow

3.2 Implementation

The dataset used is the Intellifusion 3D face data set. It contains around 400,000 faces from 1200 people, with the total size of around 36.8GB. Each face

(a) RGB Image (b) Depth Image

Fig. 3: Example Images in the Data Set[1]

contains one RGB image and one numpy array (.npy) representing the depth information of the image.

The pre-processing of the dataset including: removing the noise data, resizing the images to 182*182, normalizing the pixels, randomly cropped and combining the RGB image and the depth image into a 4-channel RGB-D image. Then, we implement a Resnet model for training the data. The model structure is shown in Figure 4.

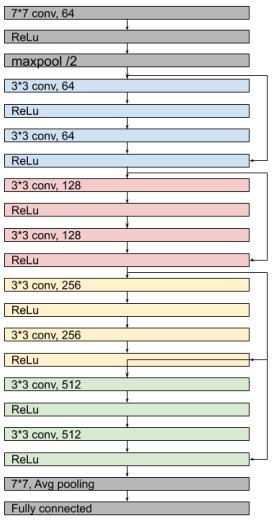

Fig. 4: structure of the network

To improve the speed of the training, we used GPU to accelerate the training [10]. The GPU used is Nvidia GeForce RTX 2080. Table 1 shows the differenct average time for each iteraction for using GPU or without using GPU.

Table 1: Time Spent Comparison

	Using GPU	Using CPU
Time (min / iteration)	~7	~102

3.3 Evaluation

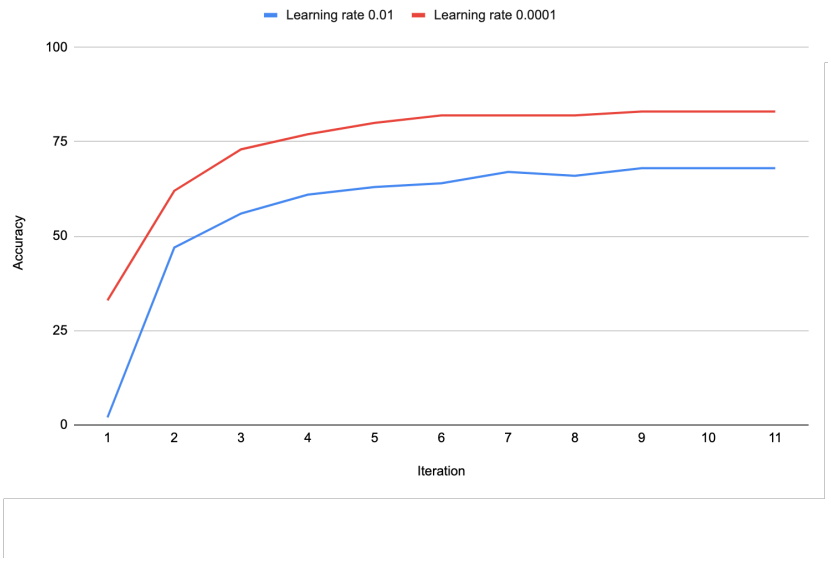

Fig. 5: Model Accuracy with Different Settings

Based on the above implementation, we did some experiment on the performance / accuracy of the ResNet model by changing some variables. Figure 5 shows the improvement of the model in different settings.

4 Conclusion

This paper implements a variant of the CNN model with residual blocks and train the model on the Intellifusion 3D face data set. By experiment, we reach the best accuracy of 82% with learning rate as 0.001.

References

1. Intellifusion 3D face data set `http://125.39.136.212:8484/3dvggface2\_1.tar.gz`.

2. Sirovich L, Kirby M. Low-dimensional procedure for the characterization of human faces. Josa a. 1987 Mar 1;4(3):519-24.
3. Turk M, Pentland A. Eigenfaces for recognition. Journal of cognitive neuroscience. 1991 Jan;3(1):71-86.
4. Amos B, Ludwiczuk B, Satyanarayanan M. Openface: A general-purpose face recognition library with mobile applications. CMU School of Computer Science. 2016 Jun;6.
5. Bradski GR. Computer vision face tracking for use in a perceptual user interface.
6. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. InAdvances in neural information processing systems 2012 (pp. 1097-1105).
7. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).
8. Jain AK, Klare B, Park U. Face recognition: Some challenges in forensics. InFace and Gesture 2011 2011 Mar 21 (pp. 726-733). IEEE.
9. Schroff F, Kalenichenko D, Philbin J. Facenet: A unified embedding for face recognition and clustering. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 815-823).
10. Bouvrie J. Notes on convolutional neural networks.
11. PyTorch CUDA semantics: `https://pytorch.org/docs/stable/notes/cuda.html`. Last accessed 27 Oct 2019
12. Vasilache N, Johnson J, Mathieu M, Chintala S, Piantino S, LeCun Y. Fast convolutional nets with fbfft: A GPU performance evaluation. arXiv preprint arXiv:1412.7580. 2014 Dec 24.
13. Li Z, Jia H, Zhang Y, Chen T, Yuan L, Cao L, Wang X. AutoFFT: a template-based FFT codes auto-generation framework for ARM and X86 CPUs. InProceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis 2019 Nov 17 (p. 25). ACM.
14. Frigo M, Johnson SG. FFTW: An adaptive software architecture for the FFT. InProceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181) 1998 May 15 (Vol. 3, pp. 1381-1384). IEEE.
15. Takahashi D. FFTE: A fast Fourier transform package. http://www. ffte. jp/. 2005.
16. Gao W, Tang F, Wang L, Zhan J, Lan C, Luo C, Huang Y, Zheng C, Dai J, Cao Z, Zheng D. AIBench: an industry standard internet service AI benchmark suite. arXiv preprint arXiv:1908.08998. 2019 Aug 13.
17. Gao W, Luo C, Wang L, Xiong X, Chen J, Hao T, Jiang Z, Fan F, Du M, Huang Y, Zhang F. AIBench: towards scalable and comprehensive datacenter AI benchmarking. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 3-9). Springer, Cham.
18. Hao T, Huang Y, Wen X, Gao W, Zhang F, Zheng C, Wang L, Ye H, Hwang K, Ren Z, Zhan J. Edge AIBench: towards comprehensive end-to-end edge computing benchmarking. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 23-30). Springer, Cham.
19. Jiang Z, Gao W, Wang L, Xiong X, Zhang Y, Wen X, Luo C, Ye H, Lu X, Zhang Y, Feng S. HPC AI500: a benchmark suite for HPC AI systems. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 10-22). Springer, Cham.

20. Luo C, Zhang F, Huang C, Xiong X, Chen J, Wang L, Gao W, Ye H, Wu T, Zhou R, Zhan J. AIoT bench: towards comprehensive benchmarking mobile and embedded device intelligence. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 31-35). Springer, Cham.
21. Hao T, Zheng Z. The Implementation and Optimization of Matrix Decomposition Based Collaborative Filtering Task on x86 Platform. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
22. Chen M, Chen T, Chen Q. An Efficient Implementation of the ALS-WR Algorithm on x86 CPUs. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
23. Deng W, Wang P, Wang J, Li C, Guo M. PSL: Exploiting Parallelism, Sparsity and Locality to Accelerate Matrix Factorization on x86 Platforms. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
24. Gao W, Tang F, Wang L, Zhan J, Lan C, Luo C, Huang Y, Zheng C, Dai J, Cao Z, Zheng D. AIBench: an industry standard internet service AI benchmark suite. arXiv preprint arXiv:1908.08998. 2019 Aug 13.
25. Gao W, Luo C, Wang L, Xiong X, Chen J, Hao T, Jiang Z, Fan F, Du M, Huang Y, Zhang F. AIBench: towards scalable and comprehensive datacenter AI benchmarking. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 3-9). Springer, Cham.
26. Hao T, Huang Y, Wen X, Gao W, Zhang F, Zheng C, Wang L, Ye H, Hwang K, Ren Z, Zhan J. Edge AIBench: towards comprehensive end-to-end edge computing benchmarking. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 23-30). Springer, Cham.
27. Jiang Z, Gao W, Wang L, Xiong X, Zhang Y, Wen X, Luo C, Ye H, Lu X, Zhang Y, Feng S. HPC AI500: a benchmark suite for HPC AI systems. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 10-22). Springer, Cham.
28. Luo C, Zhang F, Huang C, Xiong X, Chen J, Wang L, Gao W, Ye H, Wu T, Zhou R, Zhan J. AIoT bench: towards comprehensive benchmarking mobile and embedded device intelligence. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 31-35). Springer, Cham.
29. Chen M, Chen T,Chen Q. An Efficient Implementation of the ALS-WR Algorithm on x86 CPUs. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
30. Deng W, Wang P, Wang J, Li C, Guo M. PSL: Exploiting Parallelism, Sparsity and Locality to Accelerate Matrix Factorization on x86 Platforms. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
31. Xiong X, Wen X, Huang C. Improving RGB-D face recognition via transfer learning from a pretrained 2D network. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
32. Li J, Jiang Z. Performance Analysis of Cambricon MLU100. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
33. Li G, Wang X, Ma X, Liu L, Feng X. XDN: Towards Efficient Inference of Residual Neural Networks on Cambricon Chips. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
34. Hou P, Yu J, Miao Y, Tai Y, Wu Y, Zhao C. RVTensor: A light-weight neural network inference framework based on the RISC-V architecture. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.