Convolutional Neural Network for Face Recognition Based on LFW dataset

Jiaqi Wu, UFID: 09582676

***Abstract*—This project explores one kind of common used method which is Convolutional Neural Network (CNN) for face recognition. The CNN we used contains two convolution layers, two max-pooling layers and one connection layer. The activation function we used is Relu, the classifier is softmax classifier and the optimizer is Adam optimizer. The accuracy can be 89.0% on LFW dataset. So this model is feasible for face recognition.**

***Index Terms*—convolutional neural network, CNN, face recognition, LFW database**

I. Introduction

Face recognition has become one of the most studied topics in computer vision and biometrics since the 1970s. It can be widely used in security, health, marketing and retail and many other applications because of its robustness and non-invasive interaction comparing to the fingerprint recognition, iris recognition, voice recognition and many other methods.

Face recognition is a process of verifying or identifying of the given human faces. Verifying is the task to distinguish if the two faces are from one person. And identifying is the task to look in the database containing more than one person to see which person the given face belongs to in the database.

The challenging is that the faces of the same person can also vary a lot in Pose, Illumination, Expression, and etc. Also, the sizes and formats of the pictures to be detected can be different. The common idea of face recognition is to detect the faces, extract the features, then train the models.

In this project, the library OpenCV is used to preprocess the images and detect the face. This library is very convenient and useful to develop image processing, computer vision, and pattern recognition problems. Then the convolutional neural network is applied as the learning algorithm. Convolutional neural network is one of the most common used deep learning algorithms while solving the problems about images. It can learn the robust face features after being trained with a large amount of data. And we don't need to design the features and extract the features that are robust with respect to pose, illumination, expression and many other intra-class differences. These features can be learnt through learning process. But the weakness is that CNN need a large amount of data to be trained and the dataset need to contain enough variance.

.

The dataset we used is a widely used Labeled Faces in the Wild (LFW) dataset. This is an unconstrained dataset which is in large-scale. It includes 13233 face images that are assigned to 5749 different person names. The images are captured in uncontrolled environments, so they can vary a lot in pose, illumination and expression. Another disadvantage of CNN is that the training time is too long because of the large scale training dataset. In order to quickly evaluate the CNN algorithm, we just use the first 1000 images of the dataset to train and test.

II. Description

The overall design of this project is shown in Fig. 1. The images are preprocessed, then put into the neural network to train the model. The test images are also preprocessed, then put into the trained model and get the output of the test.

There are three sections in this project work: 1) Image preprocessing; 2) Convolutional neural network designing; 3) Model training and testing; 4) Evaluation

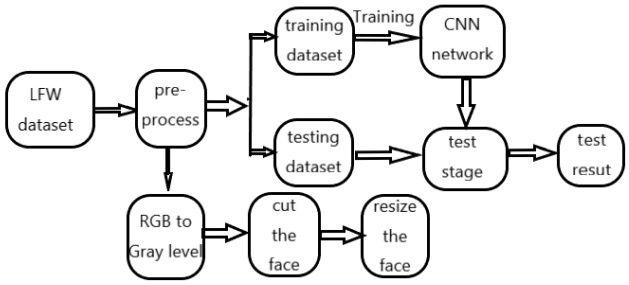

Figure 1. Overall design scheme.

A. Image Preprocessing

The images in the LFW dataset contains different background that is useless for face recognition and the noise background may also introduce redundant features which can influence the performance for neural network. So we need to

preprocess the images so that the images are suitable for training in the neural network. The OpenCV library is used.

First the color images in the dataset are read and convert into grey level images. Because the difference of RGB level of a person's face is small, the gray level is more easier and convenient for training.

The next step is to detect the face in the images. The bounding box of the face is generated for each image. Then the faces are clipped using the bounding boxes. After that, the faces are resized to the uniform size which will be the input size of the neural network. Fig. 2 shows some images before preprocessing and after preprocessing. Fig. 2 (a) shows the images before preprocessing. These images contain different background and have variance in pose, illumination, expression, and etc. Fig. 2 (b) shows the faces that are intercepted from the raw images in the LFW dataset and resized to 200*200.

Figure 2. The images from the LFW dataset before and after preprocessing. (a) Images before preprocessing; (b) Images after preprocessing.

The final step is to label these images according to their names using the binary class labels. Then split the dataset into training data and test data.

B. Convolutional Neural Network Designing

The structure of the neural network can be shown in Fig. 3. In the following neural network, there are input layer, hidden layers, and output layer. For each neuron of these layers, the output can be calculated using the following formula:

$$h_{W,b}(x) = f(\Sigma\ Wx+b) \qquad (1)$$

W is the weight vector of this neuron, and b is the bias of this neuron. The function $f()$ is the activation function of the neuron.

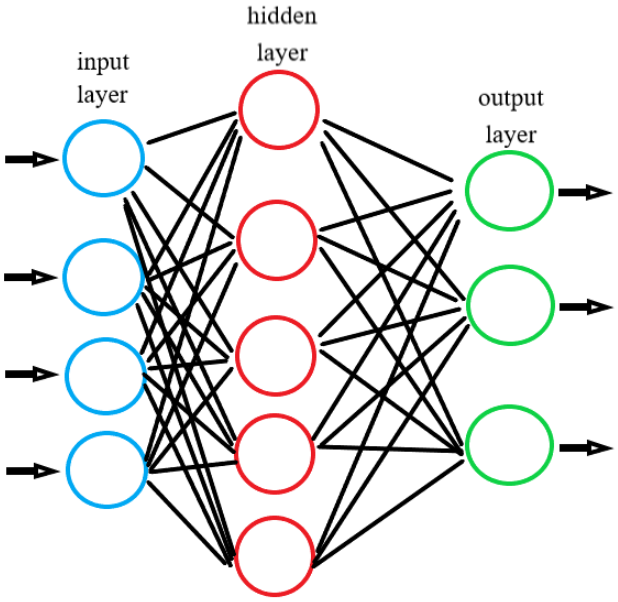

Figure 3. The structure of neural network

The convolutional neural network is a typical architecture of neural network. There are three factors that can influence the performance of the face recognition based on CNN: 1) Training dataset; 2) CNN structure; 3) Optimizer. In this project, we use LFW dataset for training. The CNN structure is shown in Fig. 6.

This structure contains 2 convolutional layers. After the images are going through the convolutional layer, the number of output convolutional features are very large. So the computational complexity is large and it's prone to over-fitting. The solution is to add a pooling layer after each convolutional layer. The pooling layer can reduce the size of output. There are three common pooling ways: 1) Mean-pooling: Average the output in the neighborhood. It performs better in keeping the background information. 2) Max-pooling: Choose the maximum value of the output in the neighborhood. It performs better in keeping the texture information. 3) Stochastic-pooling: Assign the probability to the pixel according to the value, and then do the sub-sampling according to the probability. In this task, we choose max-pooling because it can keep the texture information. We also add a dropout layer after each pooling layer to reduce the over-fitting.

We choose the Relu function as the activation function for each convolutional layer. The plot of the function is shown in Fig. 4. The function of Relu is the following formula:

$$f(x) = max(0, x) \qquad (2)$$

This activation function is much faster than sigmoid function, because its calculating cost is very small.

In the output layer, we choose the softmax function as the activation function. Because it is a common used multi-classifier which is simple and efficient. Softmax is a function that can convert a set of $(-\infty,+\infty)$ scores into a set of probabilities and normalize their sum to 1. The example is shown in Fig. 5. The input are 6 scores and the output are 6 probabilities and their sum is 100%.

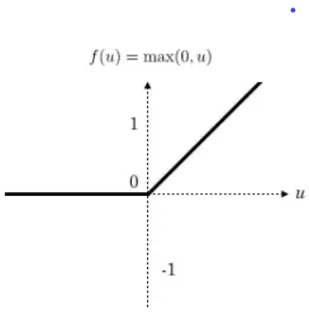

Figure 4. The Relu function.

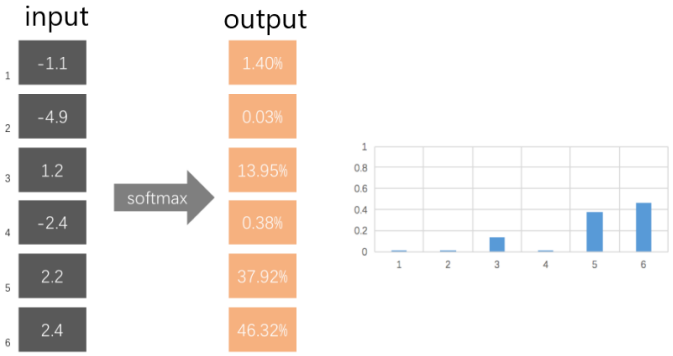

Figure 5. The example of softmax function classifier.

As shown in Fig. 6, the input images' size is 200*200, then they are put into the first convolutional layer represented by c1. The kernel size of this layer is 3*3 and the kernel number is 32. After that, the following layer is the max-pooling layer represented by p1. The pool size is 2*2 and the stride size is 2*2. Then d1 is the dropout layer. The following layer is the second convolutional layer represented by c2.The kernel size of this layer is also 3*3 and the kernel number is 64. Then the max-pooling layer p2 is followed. Its parameters are the same as p1. Then the same dropout layer is d2. After that, flatten the output and connect them using connection layer presented by con1. The output size of the connection layer is 512. Finally, the output layer is followed and the output size is the number of classes.

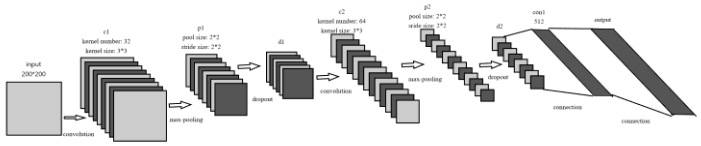

Figure 6. The structure of convolutional neural network.

C. *Model Training and Testing*

During the training process, the batch learning is used to train the model in order to reduce the influence of the noise. The batch size is set as 32. Then train the model 100 epochs.

After the model is optimized, test the model using the test dataset split from LFW dataset.

Choose different optimizers and compare the evaluation results.

III. EVALUATION

The choosing of optimizer can influence the performance of the model. So we tried different optimizer such as RMSprop, SGD, and Adam. Then evaluate their performance.

Fig. 7 shows the accuracy changing trend during the model training process when using different optimizer.

Fig. 8 shows the loss changing trend during the model training process when using the different optimizer.

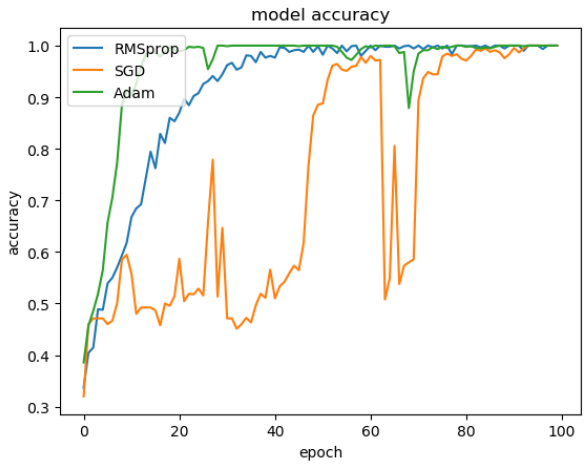

Figure 7. The accuracy of three models during the training time

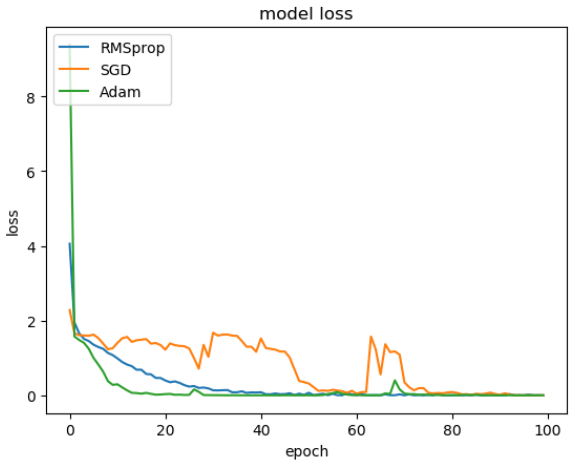

Figure 8. The loss of three models during the training time.

It can be observed in Fig. 7 and Fig. 8 that the SGD optimizer is not stable for this face recognition problem, because the accuracy and loss curves are volatile. The accuracy and the loss of RMSprop model and Adam model is comparative. But the convergence rate of Adam model is faster than RMSprop model. So Adam is a better optimizer.

The running time of training and testing of these three different models with different optimizers are shown in Fig.9. In this figure, it shows that the training time of SGD model is the minimum, and the training time of RMSprop model and Adam model is similar. The testing time of RMSprop model is the maximum, and the testing time of SGD model and Adam model is similar.

Considering the running time and the convergence condition, Adan is the best choice of the optimizer.

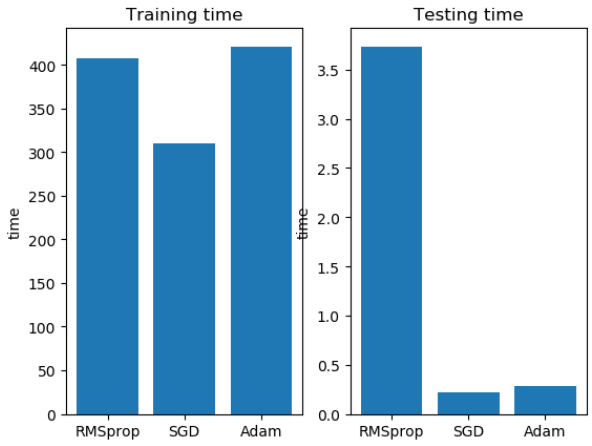

Figure 9. The running time of the models with different optimizer.

When choosing the Adam as the optimizer, the accuracy is very low during the testing process. The reason is that some classes have only 1 image. If we split the dataset and assign the only image to testing dataset, there will be no images to do the training for this class, so the training work with respect to this class is not enough.

To overcome this problem, we filter out the images in the LFW dataset. We only use the classes that the number of images in the class is more than a specific threshold. Then train the model and test the model. Table. 1 shows the accuracy and the loss of the testing with different filtering threshold.

Threshold	Accuracy	Loss
1	19.0%	35.082
5	56.0%	3.651
30	82.9%	0.841
100	89.0%	0.351

Table 1. The accuracy and the loss in the testing process with different filtering threshold.

In this table, we can see that the threshold of 30 will lead to more accuracy or smaller loss than the threshold of 1 and 5. So if we need the accuracy to be larger or the loss to be smaller, the number of images used for training should be larger. This can represent a drawback of the deep learning: large amount of dataset is needed to train the model.

.

IV. Related Work

This project is just a simple application of convolutional neural network on face recognition. In the last 30 years, deep learning method has been widely studied and applied for face recognition. And the accuracy has been boosted to 99.80% in just three years.

At first, AlexNet in 2012 is the breakthrough for the simpler problem of image classification. Then there are four milestone systems on deep learning for face recognition: DeepFace, FaceNet, VGGFace, and the DeepID series of systems.

DeepFace is based on convolutional neural network and is the major leap forward using deep learning. FaceNet introduced an innovation called '*triplet loss*' to encoded the images as feature vectors efficiently. VGGFace focus on how to assemble a very large scale dataset by a combination of automation and human in the loop. DeepID can increase the inter-personal variations and reduce the intra-personal variations.

Now people are still work on computer vision and progress will continue.

V. Summary and Conclusion

This project applied a common used and useful method called convolutional neural network for face recognition problem. In this project, the dataset is Labeled Faces in the Wild (LFW) dataset which is widely used. And the goal is to train the model using dataset picked from LFW dataset and identify the name of a given image. So this is a closed-set identification task.

In this task, two convolutional layers are built and the best optimizer is the Adam optimizer. The accuracy of the model

can be up to 89% and if more images are trained in one class, the accuracy will keep improving.

REFERENCES

[1] I. Masi, Y. Wu, T. Hassner and P. Natarajan, "Deep Face Recognition: A Survey," 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Parana, 2018, pp. 471-478.

[2] Yin, Xi & Liu, Xiaoming. (2017). Multi-Task Convolutional Neural Network for Face Recognition. IEEE Transactions on Image Processing. PP. 10.1109/TIP.2017.2765830.

[3] Tt, Tt & Mohamad, Dzulkifli. (2007). A Summary of literature review: Face Recognition. Postgraduate Annual Research Seminar. 2007. 3-4.

[4] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Crossref. Web.

[5] Chaochao Lu and Xiaoou Tang. 2015. Surpassing human-level face verification performance on LFW with gaussian face. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15). AAAI Press, 3811–3819.

[6] S. Liao, Zhen Lei, Dong Yi and S. Z. Li, "A benchmark study of large-scale unconstrained face recognition," IEEE International Joint Conference on Biometrics, Clearwater, FL, 2014, pp. 1-8.

[7] Viola, P., Jones, M.J. Robust Real-Time Face Detection. *International Journal of Computer Vision* **57,** 137–154 (2004).

[8] Coşkun, Musab & Uçar, Ayşegül & yıldırım, Özal & Demir, Yakup. (2017). Face Recognition Based on Convolutional Neural Network.. 10.1109/MEES.2017.8248937.