International Journal of Future Generation Communication and Networking
Vol. 13, No. 2s, (2020), pp. 1299–1305

# Multiplefaces Detection And Recognition Using Machine Learning

Nikita Thadani[1],Pranjal Dhore[2],Nazneen Tarannum Wasim Ahmad Khan[3]

[1]*Mtech Student, Computer Science & Engineering Department, JIT Nagpur, India*
[2]*Assistant Professor, Computer Science & Engineering Department, JIT Nagpur, India*
[3]*Assistant Professor, Computer Science & Engineering Department, JIT Nagpur, India*

**Abstract**

*A complete face recognition system includes face detection, face preprocessing and face recognition processes. Therefore, it is necessary to extract the face region from the face detection process and separate the face from the background pattern, which provides the basis for the subsequent extraction of the face difference features. Face recognition of the separated faces is a process of feature extraction and contrast identification of the normalized face images in order to obtain the identity of human faces in the images. The aim of this paper is to analyze the present research results of face recognition technology, and studies a face recognition algorithm.*

***Index Terms:*** *Face detection, Machine Learning, Feature Extraction, convolutional neural network*

## 1. INTRODUCTION

Convolutional neural network (CNN) is a deformation of multi-layer perceptron inspired by biological vision and the most simplified pre -processing operation. It is essentially a forward feedback neural network. The biggest difference between convolutional neural network and multi-layer perceptron is network. The first few layers are composed of a convolutional layerand a pooled layer alternately cascaded to simulate a simple cascade of cells and complex cells for high-level feature extraction in the visual cortex. At present, face recognitionalgorithms can be roughly divided into two categories:
(1) Representation-based methods. The basic idea is to convert two-dimensional face input into another space, and then use statistical methods to analyze face patterns, such as Eigenface, Fisherface, and SVM.
(2)A feature-based method generally extracts local or global features and then sends a classifier for face recognition, such as recognition based on set features and HMM.
The Siamese network is neural network for measuring of similarity. It can be used for category identification, classification, etc., in the scenario when there are many categories, but the number of samples per category is small. The traditional classification method for distinguishing is to know exactly which class each sample belongs to and need to have an exact label for each sample. And the relative number of tags is not too much. These methods are less applicable when the number of categories is too large and the number of samples per category is relatively small. In fact, it is also very well understood. For the entire data set, our data volume is available, but for each category, there can be only a few samples, then using theclassification algorithm to do it, because each category of samples is too Less, we can't train any good results at all, so we can only find a new way to train this data set, thus proposing the Siamese network, as show in below figure

ISSN: 2233-7857 IJFGCN

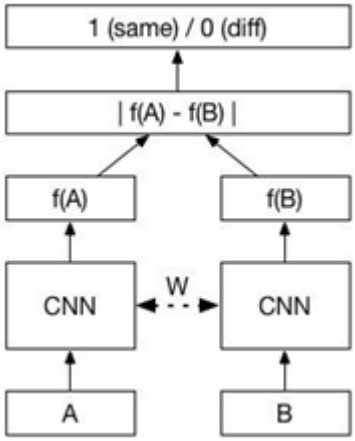



Figure: 1.1Siamese Network Workflow

## TRAIN THE NEURAL NETWORK

The face database we choose is ORL [15]. The ORL face database consists of 400 pictures of 40 people, that is, 10 pictures per person. The face has expressions, tiny gestures and so on. The training processing is performed on the two databases, and 90% of the faces in the library are randomly selected as the training set, and the remaining 10% of the faces are used as test sets, and then the faces in the two sets are standardized. The training process was using GPU. We can see during training, the CPU usage went to 100%, and the working temperature increased dramatically.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.48                 Driver Version: 390.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1050    Off  | 00000000:01:00.0 Off |                  N/A |
| N/A   35C    P8    N/A /  N/A |    147MiB /  4042MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1037      G   /usr/lib/xorg/Xorg                            78MiB |
|    0      1223      G   /usr/bin/gnome-shell                          68MiB |
+-----------------------------------------------------------------------------+
```

Figure:1.1 (a) GPU Usage before training

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.48                 Driver Version: 390.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1050    Off  | 00000000:01:00.0 Off |                  N/A |
| N/A   59C    P0    N/A /  N/A |   1972MiB /  4042MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1037      G   /usr/lib/xorg/Xorg                            78MiB |
|    0      1223      G   /usr/bin/gnome-shell                          68MiB |
|    0      2206      C   /home/u18/anaconda3/envs/fastai/bin/python  1815MiB |
+-----------------------------------------------------------------------------+
```

Figure: 1.1.(b) GPU Usage during training.

ISSN: 2233-7857 IJFGCN

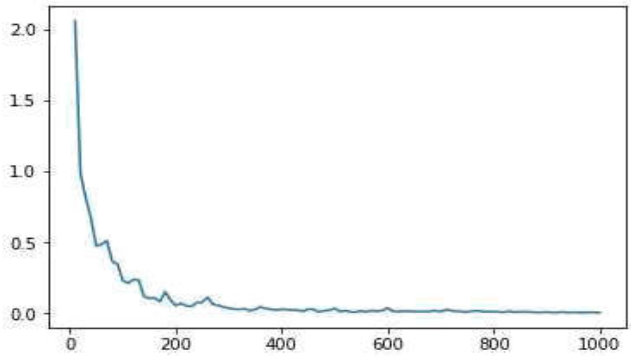



Figure:1.1 (c) Training loss vs. Epoch

## 2. MODEL VERIFICATION

To calculate if the faces on two images come from the same person, we need to calculate the similarity of the two images, aka, the Euclidean distance between two vectors. Below is an output of two dis-similar people identified by the model:


Figure: 2. Different people with high Euclidean distance

Vector of Face 1: Variable containing:

1.7350 0.2165 1.0214 1.5764 2.2253
[torch.cuda.FloatTensor of size 1x5 (GPU 0)] Vector of
Face 2: Variable containing: -0.7570 1.5081 0.3380
1.5524 -0.0977

[torch.cuda.FloatTensor of size 1x5 (GPU 0)]
Distance between Face1 Vector and Face2 Vector: Variable containing:
3.7070
[torch.cuda.FloatTensor of size 1x1 (GPU 0)]

ISSN: 2233-7857 IJFGCN

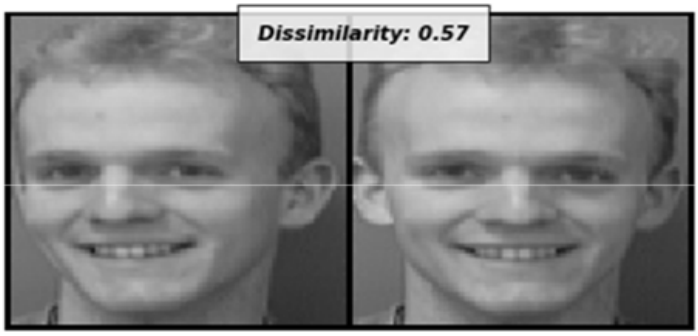



Figure: 2.1. Same person, even with different pose, has small Euclidean distance

This is an example output of similar person with different pose, identified by our model, and the images are shown in above fig.

Vector of Face 1: Variable containing:
1.7350 0.2165 1.0214 1.5764 2.2253

[torch.cuda.FloatTensor of size 1x5 (GPU 0)]

Vector of Face 2: Variable containing:
1.6301 0.7585 1.1658 1.6345 2.2486 [torch.cuda.FloatTensor of size 1x5 (GPU 0)]

Distance between Face1 Vector and Face2 Vector: Variable containing:
0.5741

[torch.cuda.FloatTensor of size 1x1 (GPU 0)]

## 3. EXPERIMENTAL RESULTS

The experimental results in this paper can be divided into three parts: performance of tracking, recognition rate for multiple and different faces, recognition rate and time taken to process on CUFace dataset.

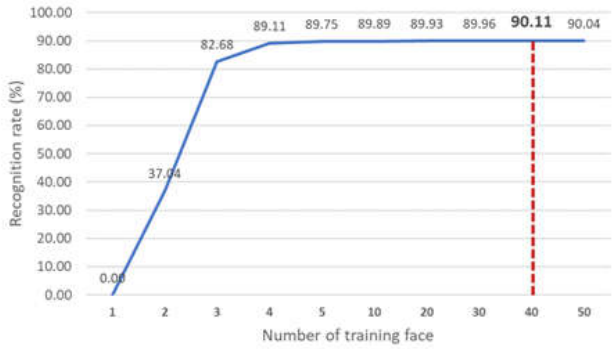
### 1. Tracking performance

To show the tracking algorithm performance, we tested the detection algorithm with YoutubeFace database which contains the video clip of people movement. The tracking algorithm performance is comparable to normal detection where the time taken to process is faster. The results are shown in Table 1.

Table 1. Comparison of average precision, recall and f-score of detection and detection with tracking.

| | Precision | Recall | F-Score |
|---|---|---|---|
| Detection | 0.75 | 0.82 | 0.76 |
| Detection with tracking | 0.69 | 0.99 | 0.75 |

ISSN: 2233-7857 IJFGCN




## 2. Recognition rate for multiple faces

The recognition rate depends on the number of trained faces. We test the algorithm with FEI database. The test set consists of people face images in different direction which is difficult to recognize when the features of face does not appear. The result in Figure 1 indicates that 40 trained face images achieves the highest recognition rate.

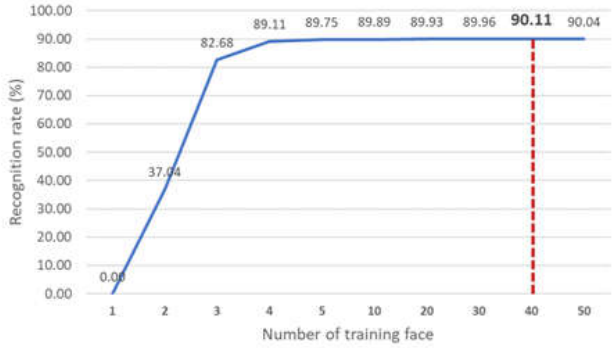



Figure.3.Comparison of recognition rate and number of training face on FEI database.

## 3. Recognition rate and processing time

To measure recognition rate and the processing time of recognition algorithm of different and many faces, we tested the algorithm with CUFace dataset. We compared normal recognition and recognition using tracking algorithm. The processing time is speeded up by 56.10% on average. The frame rate varied by the number of face in recognition, our algorithm can recognize multiple face around 5-10 fps with recognition rate around 90%.

## 4. CONCLUSIONS

In this paper, we proposed a framework for real-time multiple face recognition.The recognition algorithm based on CNN which is the state-of-art algorithm. The framework consists of the tracking technique and using the minimal weight of the model. This can reduce the processing time and network parameter to learn recognize multiple face feature in real-time.

We proposed to build a high performance, scalable, agile, and low cost face recognition system. We divide the proposed approach into several small sub projects. First, we studied neural network and convolutional neural network. Based on the theory of deep learning, we built the Siamese network which will train the neural network based on similarities. Then we examine and compare the available open source data set, we chose ORL dataset and trained the model with GPU. The model will take a human face image and extract it into a vector. Then the difference between vectors are compared to determine if two faces on different picture belongs to the same person.

ISSN: 2233-7857 IJFGCN

Then we did the study, compare, design and build a system to work with the neural network model. The system uses client-server architecture. GPU is used on the server side to provide high performance. We also de-coupled the main components of the system to make it flexible and scalable. We used the non-block and asynchronies features of Node.JS to increase the system's concurrency. Since the entire system is modularized, it can be used in different domains, thus reduced the development cost.

When build the neural network model, there are many parameters which can be tuned to increase the model performance. Wecan keep tuning our models to increase its accuracy.Also, for a trained base model, we can re-train it usinga specific dataset.So another way to increase the whole system's performance is to capture the specific people's images and re-train the model based on this small dataset. For example, if an organization with 3000 people uses this system, the model can be trained to be very accurate on these 3000 people. We can employ and automate this feature into the system.Also, we can apply implementation of face recognition system for recognizing faces and also calculating or predicting the age and expression.

## REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815-823.
[2] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXivpreprint arXiv:1502.00873,* 2015.
[3] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conferenceon computer vision and pattern recognition*, 2014, pp.1701-1708.
[4] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision,* vol. 57, no. 2, pp. 137-154, 2004.
[5] "cookbook.fortinet.com," 10 10 2018. [Online]. Available: https://cookbook.fortinet.com/face-recognition-configuration-in-forticentral/. [Accessed 10 10 2018].
[6] M. J. Paul Viola, "Robust Real-time Object Detection," International Journal of Computer Vision, pp. 137-154, 2004.
[7] H. A. C. H. a. S. L. Bo Wu, "Fast rotation invariant multi-view face detection based on real Adaboost," Sixth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 79-84, 2004.
[8] H. A. Y. T. S. L. Yuan Li, "Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Life Spans," IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1728-1740, 2008.
[9] C. J. T. D. H. C. A. J. G. T. F. COOTES, "Active Shape Models-Their Training and Application," COMPUTER VISION AND IMAGE UNDERSTANDING, pp. 38-59, 1995.
[10] D. C. a. T. Cootes, Boosted Regression Active Shape Models, BMVC, 2007.
[11] Y. W. F. W. a. J. S. X. Cao, "Face alignment by Explicit Shape Regression," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, 2012.
[12] M. T. a. A. Pentland, "Eigenfaces for recognition," Journal of Cognitive Neuroscience, pp. 71-86, 1991.
[13] S. S. W. G. X. C. a. H. Z. Wenchao Zhang, "Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition," Tenth IEEE International Conference on Computer Vision, pp. 786-791, 2005.
[14] X. C. F. W. a. J. S. D. Chen, "Blessing of Dimensionality: High-Dimensional Feature and Its Efficient Compression for Face Verification," in 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, 2013.
[15] "Convolutional_neural_network," 2017. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network.

ISSN: 2233-7857 IJFGCN

[16] S. S. Liew, "Research Gate," 1 3 2016. [Online]. Available: https://www.researchgate.net/figure/Architecture-of-the-classical-LeNet-5-CNN_fig2_299593011. [Accessed 10 10 2018].
[17] L. B. Y. B. a. P. H. Y. LeCun, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, pp. 1-45, 11 1998.
[18] xlvector,"JIANSHU," 25 7 2016. [Online]. Available: https://www.jianshu.com/p/70a66c8f73d3. [Accessed 18 9 2018].
[19] F. S. Samaria, Face recognition using Hidden Markov Models, Doctoral thesis, 1995.
[20] E. H. G. R. A. L. H. Learned-Miller, " Labeled Faces in the Wild: A Survey," Advances in Face Detection and Facial Image Analysis, pp. 189-248, 2016

ISSN: 2233-7857 IJFGCN