Face Recognition With Database Face Verification A Comparison between Template Matching and Pretrained Convolutional Neural Network

Shi Bin Teo
School of Computer Science
University of Nottingham
Nottingham, UK
hcyst2@nottingham.ac.uk

***Abstract*—This work compares the face recognition capability between template matching and deep convolution neural network. Extra implementation regarding the implementation for face identification is also being discussed in this work. Time, face verification and face identification are evaluated and compared among models.**

***Index Terms*—face recognition, face verification, face identification**

I. Introduction

Face Recognition is a vast field that serve astonishing automation for diverse application. Face recognition methods have already been developed into hundreds of dedicated applications and that include camera surveillance, attendance system [1], face mask detection and more. Recent studies of face recognition has shifted towards more complex and fine grained solutions as it begins to reach maximum performance, some have suggested a completely new class of neural network layer and some have developed novel loss functions [2]–[4]. In fact, there a so many loss functions developed just for this specific domain and a comparative analysis shows that the triplet loss that we will be using here is actually not the best loss function. [5]

However, there are still a lot of problems to be solved in order to achieve higher performance for general-purpose face recognition, recent survey has categorized face recognition (FR) problems into 4 classes cross-factor FR, Heterogeneous FR, Multi-Single Media FR, FR in industry. [6] All of them are crucial and targeted to improve the robustness of the model. All these problems are still ongoing research as they are being addressed separately.

The goal of this work is to recreate a face recognition model also known as face identification that identify a face given an image and match it to the database. The identification is done using face verification to classify similar and dissimilar faces. A simple baseline is added to visualize how big of a difference between advance method and standard method. This report will focus mostly on the implementation of the identification system while giving some thoughts in different problems that arise.

II. Methodology

A. Problem Statement

The goal is to label each of the test image to an existing face with their ID within the train database given both train and test set of images. One extra constraint on testing set is that it will contain only existing faces that can be found within the train database, therefore the labelling of no valid face or no match face is not required.

B. Method 1 Template Matching

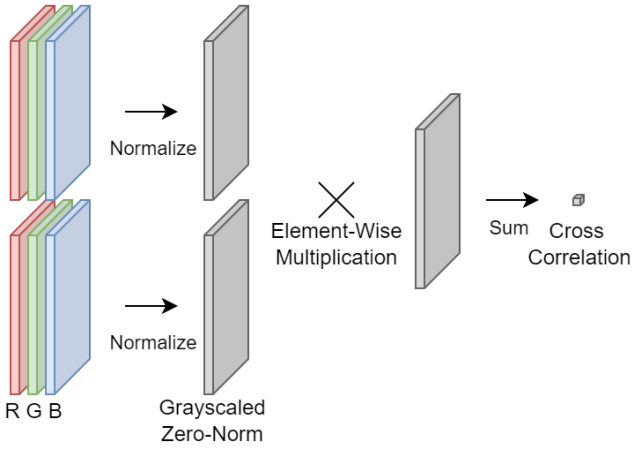

Fig. 1. **Zero-Normalized Cross-Correlation Workflow.** Faces are verified using highly correlated intensity values

$$\frac{1}{n}\sum_{x,y}\frac{1}{\sigma_f\sigma_t}(f(x,y)-\mu_f)(t(x,y)-\mu_t) \tag{1}$$

The baseline method is Zero-normalized cross-correlation based template matching depicted in Fig. 1, usually it is designed to slide a smaller size test template across the train images to find high correlation region, however since no extra face detection is added for the testing image, the bounding box of the face within the test image is unknown, therefore the whole test image is directly used as a template to compute

the correlation values for each train image within the database. All images are first processed into grey images by averaging their channels and then normalized by subtracting its mean and divided by its standard deviation (1). Finally, correlation values are calculated using simple matrix multiplication for the pair of normalized feature vectors. The test image is then labelled with the highest correlated train image label.

C. Method 2 Deep Convolutional Neural Networks

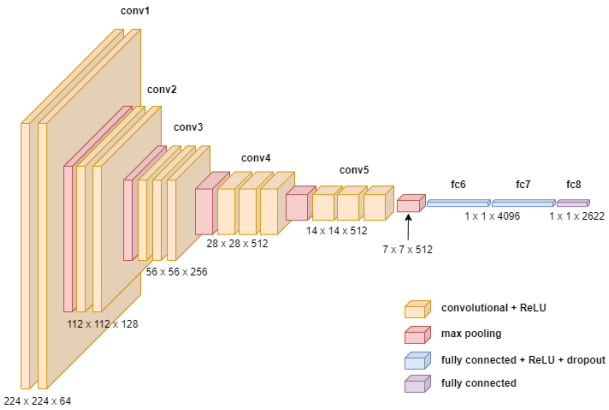

Fig. 2. **VGGFace Architecture.** 224 by 224 RGB image is fowarded into the model and embedded feature vector that lengths 2622 will be the output. Each layers are presented along with its input size but some layers share the same dimensions. All convolutional layers used 1 pixel of zero padding and then max pooled with a kernel size of 2 by 2 and a stride of 2. Fc layers that contain dropout are configured with a dropout rate of 0.5

The next method is a pretrained deep convolutional neural network (DCNN) called VGGFace is used to embed images into deep feature representations which can be used for measuring the differences directly within the distance space. [7] Fig. 2, the architecture of it are mostly repeating blocks of convolutional layers followed by max pooling layer and ReLU layer. In their work, they stated that the model was first trained as a smaller architecture A and deep transferred to a larger architecture B, Fig. 2, those architecture was first introduced in earlier work [8] and the training was done with the last layer being N-ways classification layer and was removed after training to reveal the feature embedding layers. The model was trained using triplet loss function where each triplet of images requires an anchor, a similar and dissimilar image. The dataset they have gathered is no longer retrievable as it was stored using URL redirect links to individual images and most images no longer exist online. Nevertheless, the pretrained parameters can be found online. The model is re-implemented in MATLAB using pretrained parameters and with that in mind the remaining implementation is to improve identification performance.

$$euclidean(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{2}$$

$$L2(a) = \frac{a}{\sqrt{\sum_{i=1}^{n} a_i^2}}$$

$$euclideanL2(a,b) = euclidean(L2(a), L2(b)) \tag{3}$$

$$cosine(a,b) = 1 - \frac{\sum_{i=1}^{n}(a_i * b_i)}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{4}$$

The identification is done by doing verification using the testing image which is the face that we want to find and compute the distance to each training images within the database. Due to the fact that database images will always be re-embedded every time we match an image to it, a precompute function is written to pre-embed database images into their feature representation and can be loaded easily as they are much smaller in size compare to image. The two embedding is then being used to compute the similarity distance. 3 similarity metrics are implemented to verify faces, Euclidean distance (2), L2 normalized Euclidean distance (3) and inverted cosine similarity (4). Note that a and b are the feature vectors of two embedded images. Ultimately, the label is assigned using the smallest distanced train image label. This model can be considered as a late merging Siamese network, where two inputs are processed by two identical sister networks that embed images and calculates a distance value at the final layer.

III. Evaluation

A. Identification Speed Performance

TABLE I
PRECOMPUTE SPEED TEST COMPARISON BY IDENTIFYING 1344 TESTING PROBE IMAGES ON 100 TRAINING DATABASE SUBJECTS. THE FIRST COLUMN DENOTES THE ELAPSE TIME WITHOUT PRECOMPUTE FUNCTION AND THE FOLLOWING COLUMNS DENOTES THE REQUIRED TIME USING PRECOMPUTED DATABASE

Models	Total Time (s)	
	without	***precomputed***
Template Matching	33.0650	-
VGGFace Euc.	569.0000	262.8354
VGGFace Euc. L2	641.1500	264.6170
VGGFace Cos.	589.3315	265.1377

all VGGFace models have the same database embedding time which is 26.2873s and it is not included in precomputed time

Additional code optimizations are also done to improve run-time performance of the identification system, first, as mentioned before the train database is precomputed before verifying faces and it significantly reduces the time required by 50%. In Table I, the result is obtained using CPU and the calculation was done using batch size of 64 and averaged after 10 times of running. The general idea of this is to check if it can improve the performance. However, the score can

differ on different computer setup. This is implemented to improve usability and make it as a realistic as possible with the assumption of the user already been registered into the database and trying to verify his face on runtime. This simply means that we can only pre-embed database images as testing images are acquired on runtime.

B. Face Verification Performance

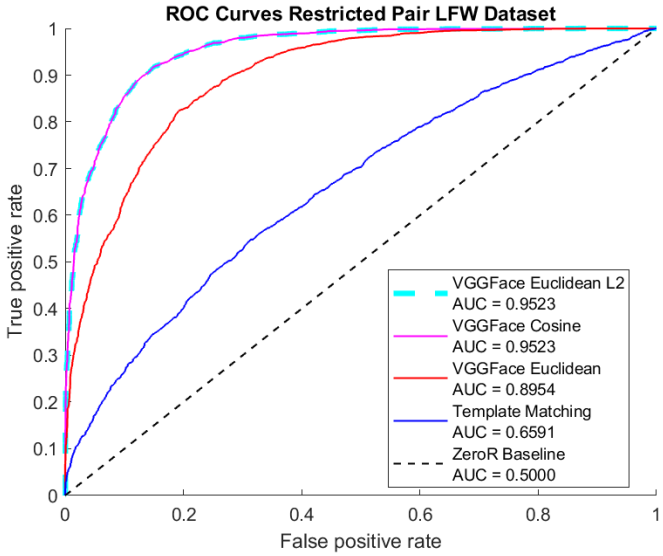

Fig. 3. **Face Verification ROC Curve.** This indicates how well the model verify similar and dissimilar faces. The test was done on LFW restricted configuration pairs, 3000 similar and 3000 dissimilar pairs were used to compute this ROC graph. Euc. L2 and Cosine methods both share the same result, then standard Euc. is the second best and template matching performs badly. The worst scenario here is to guess the same major class which is called zeroR baseline and since we have balanced classes here the AUC will always be 0.5

An extra dataset is used here as ROC requires pairs of images to be generated, thus instead of using the prepared train and test images, LFW dataset is used. LFW restricted pair configuration is chosen here because it is a balanced set that contains equal amount of similar and dissimilar faces, if unrestricted pair were used, most often we will get high amount of easily identified dissimilar pairs which will skew the curve even more towards the top left corner and making it hard to analyse. In Fig. 3, both Euclidean L2 and Cosine based distance metrics perform the best. The problem with template matching is that, it is highly dependent on quality of the face within the image, slight changes can cause it to fail completely. Convolution based solution are more robust towards slight changes and since this model is trained on triplet loss function it is more capable towards detecting the differences of faces even when dissimilar face that looks similar is given.

C. Face Identification performance

In Table II, We can clearly see that VGGFace with distance metrics of L2 normalized Euclidean distance and inverted cosine similarity perform the best with the prepared training and testing set.

TABLE II
MODELS RECOGNITION RATE BY IDENTIFYING 1344 TESTING PROBE IMAGES ON 100 TRAINING DATABASE SUBJECTS. THE ACCURACY COLUMN DENOTES THE PROPORTION OF CORRECTLY IDENTIFIED FACES OUT OF ALL TESTING FACES

Models	***Accuracy***
Template Matching	25.3720
VGGFace Euc.	85.2679
VGGFace Euc. L2	90.6250
VGGFace Cos.	90.6250

IV. DISCUSSION

Multiple Results are indicating that both VGGFace with L2 Norm Euclidean distance and cosine distance perform better in identification and verification task. However, the execution time of L2 norm Euclidean distance is slightly faster than Cosine distance, it is possible that it could be noise, for now, L2 norm Euclidean will be considered the best of all. There are more towards the testing of a face recognition model as there are so much variables for faces. Testing against age, pose, make up, ethnicity, spoofing etc. can still be done in order to achieve maximum robustness.

V. CONCLUSION

This work have briefly gone through the selected dataset and methodologies used to recognize faces. In the other hand, different metric and measures were used to compare performances from multiple aspects of recognition models. The comparison among speed, verification and identification have successfully revealed the effectiveness of these models. In conclusion, VGGFace with L2 Norm Euclidean distance metric is considered the best.

REFERENCES

[1] M. Arsenovic, S. Sladojevic, A. Anderla, and D. Stefanovic, "FaceTime — Deep learning based face recognition attendance system," in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 000053–000058, Sept. 2017. ISSN: 1949-0488.

[2] G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. Zafeiriou, "Deep Polynomial Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. arXiv: 2006.13026 version: 2.

[3] I. Kim, S. Han, S.-J. Park, J.-w. Baek, J. Shin, J.-J. Han, and C. Choi, "DiscFace: Minimum Discrepancy Learning for Deep Face Recognition," in *Computer Vision – ACCV 2020* (H. Ishikawa, C.-L. Liu, T. Pajdla, and J. Shi, eds.), vol. 12626, pp. 358–374, Cham: Springer International Publishing, 2021. Series Title: Lecture Notes in Computer Science.

[4] P. Terhörst, M. Ihlefeld, M. Huber, N. Damer, F. Kirchbuchner, K. Raja, and A. Kuijper, "QMagFace: Simple and Accurate Quality-Aware Face Recognition," *arXiv:2111.13475 [cs]*, Mar. 2022. arXiv: 2111.13475 version: 3.

[5] A. Pathak and R. Maheshwari, "Comparative Analysis of Different Loss Functions for Deep Face Recognition," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI 2019, (New York, NY, USA), pp. 390–397, Association for Computing Machinery, Dec. 2019.

[6] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, Mar. 2021.

[7] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," in *Procedings of the British Machine Vision Conference 2015*, (Swansea), pp. 41.1–41.12, British Machine Vision Association, 2015.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015. arXiv: 1409.1556.