LightFace: A Hybrid Deep Face Recognition Framework

Sefik Ilkin Serengil
Research and Development Center
Softtech A.S.
Istanbul, Turkey
sefik.serengil@softtech.com.tr

Alper Ozpinar
Mechatronics Engineering Department
Istanbul Ticaret University
Istanbul, Turkey
aozpinar@ticaret.edu.tr

***Abstract*— Face recognition constitutes a relatively a popular area which has emerged from the rulers of the social media to top universities in the world. Those frontiers and rule makers recently designed deep learning based custom face recognition models. A modern face recognition pipeline consists of four common stages: detecting, alignment, representation and verification. However, face recognition studies mainly mention the representation stage of a pipeline. In this paper, first of all a review face recognition has been done and then the description of the developed lightweight hybrid high performance face recognition framework has been made. Its hybrid feature enables to switch face recognition models among state-of-the-art ones.**

Keywords—deep learning; machine learning; face recognition; computer vision; software framework.

I. INTRODUCTION

For years of evolution, one of the key improvements of humans what makes them being as the dominant race to the other species is their ability to be socialize and living in communities by knowing, recognizing and communicating. Since artificial intelligence aims to imitate human behavior and human movements such as speaking, listening, recognizing and understanding, researchers focus on these topics hard. Human face and voice recognition studies started mostly after 90's with signal processing and mathematical representation of voice and face with different techniques [1]. Use of neural networks and vector representation directed the research to be more human like vision systems as in our brains [2]. In the literature there are also lots of pattern recognition [3] and statistical approaches available starting from those years expanding rapidly to nowadays [4], [5]. Various datasets available for face recognition and identification with simple datasets to complicated datasets already reviewed in the papers [6] [7]. Various major theoretical and conceptual frameworks exist for face verification, face identification and face recognition. With the growing of Web 2.0 and social networks the data become the new oil in the last decade and as a consequence there has been a growing interest for digging the data for valuable information. The field has gradually broadened as with huge computational power in the cloud as well as the development of GPU's for Deep Learning especially for image processing [8] [9], [10].

Also with the growing interest for following the people in the public with in the surveillance cams aiming to extract the social distance and contamination between people in during pandemic days [11], [12]. Face recognition becoming more popular and socially important for public health.

This research area constitutes a relatively a popular area which has emerged from the rulers of the social media such as Facebook and Google to the top universities in the world like University of Oxford, Carnegie Mellon University and The Chinese University of Hong Kong. Those frontiers and rule makers recently designed deep learning based custom face recognition models and adopted in their systems. A modern face recognition pipeline consists of four common stages: detecting, alignment, representation and verification [13]. However, face recognition studies mainly mention the representation stage of a pipeline which the mathematical background started years ago as mentioned before. Besides, from the academic perspective researchers of Oxford University and Carnegie Mellon University prefer to share both network structures and pre-trained weights where from the commercial perspective Google and Facebook just share the network structures. Luckily, there is a promising growth of open source community [14], [15] who trained these known models from scratch and share pre-trained weights to the public.

In this paper, first of all a review face recognition has been done and then the description of the developed lightweight face recognition framework has been made. Due to its widespread use and intensive choice as a machine learning programming language; Python was selected for the development of framework published also as open source package under MIT license [16]. Proposed and published software library package wraps the most popular face recognition models: VGG-Face [17], FaceNet [18], OpenFace [19], DeepFace [13], DeepID [20] [21] and Dlib [22]. Even though some out-of-box face recognition packages exist in the market such as face-recognition [23], there is no software package wrapping all of those state-of-the-art face recognition models yet. This study gathers those common models under a single roof. In this way, face recognition tasks can be done very easily and quickly. Besides, the development of the hybrid approach enables to switch among those state-of-the-art face recognition models in production. It also handles all common stages of a modern face recognition pipeline in the background. This feature does not exist in the software packages in the market as well.

II. FACE RECOGNITION PIPELINE

As mentioned before a modern face recognition pipeline consists of four common stages: detect, align, represent and verify. The framework we propose handles all of those stages in the background. In the following section those modules will be mentioned.

978-1-7281-9136-2/20/$31.00 © 2020 IEEE

A. Face Detection and Face Alignment

The early stages of a face recognition pipeline is detection and alignment [24]. Herein, OpenCV [25] library handles to detect frontal faces and eyes in a pretty way. That's why, detection module directly wraps OpenCV's detection model. However, OpenCV does not offer face alignment as an out-of-the-box function.

Google researchers declared that including the face alignment algorithms to the preprocessing of the source images increases the final face recognition model accuracy from 98.87% to 99.63% [17]. This is a significant improvement without the interference of the classification models.

In order to implement alignment module it is required to apply some trigonometric tricks to align faces properly. If the location of eyes already known similar to Fig. 1, then a right-angled triangle can be drawn. One side of the triangle must be horizontal. In this way, the length of the 3 sides can be known. Thus, the angle between the hypotenuse and the horizontal side can be calculated by applying the cosine rule in Formula (1). And then inverse trigonometric function used to retrieve angle A; since cosine rule returns cosine value. OpenCV already offers an eye detection module. Finding the angle value with cosine rule is easy. Then the base image will be rotated A degrees.

$$\cos A = (b^2 + c^2 - a^2)/(2bc) \quad (1)$$

Fig. 1. Face alignment

B. Representation

Face recognition models are regular convolutional neural networks (CNN). They are responsible for representing face images as vectors. A CNN is initially trained to classify identities and then its early layers will extract raw face representation feature vector [13]. In this way, it can verify faces it haven't seen before. Table I shows input output shape of those face recognition models supported in our framework. Output shapes express the number of dimensions the input image will be represented.

TABLE I. INPUT OUTPUT SHAPE OF MODELS

Model	Input Shape	Output Shape
VGG-Face	224 x 224 x 3	2622
FaceNet	160 x 160 x 3	128
OpenFace	96 x 96 x 3	128
DeepFace	152 x 152 x 3	4096
DeepID2	55 x 47 x 3	160
Dlib	150 x 150 x 3	128

Firstly, VGG-Face has the same structure of regular VGG-19 model appears in imagenet [26] competition. Researchers tuned the model from scratch for facial images. Secondly, FaceNet uses Inception [27] model structure. Thirdly, OpenFace and Dlib are designed on Inception-Resnet-V1 [28]. Finally, DeepID and DeepFace have their own unique design.

C. Verification

Representation stage represents facial images as vectors. Representation module feeds two same dimensional vectors to verification module. We will consider similarities of these two vectors in this layer. Herein, we can calculate cosine similarity, Euclidean distance [30] or its L2 form to find distances between vectors.

Euclidean distance between two vectors is the length of the line connecting them specified in the Formula (2).

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (2)$$

Multidimensional vector is very abstract concept because we cannot visualize it in a 3D space. We could still visualize a multidimensional vector to make it concrete. Fig. 2 shows the VGG-Face representations of two images. Remember that VGG-Face represent images as 2622 dimensional vectors. There are 2622 slots in bar right to the image. Each slot represents a dimension in the output vector. Its value specifies the color of slot as well.

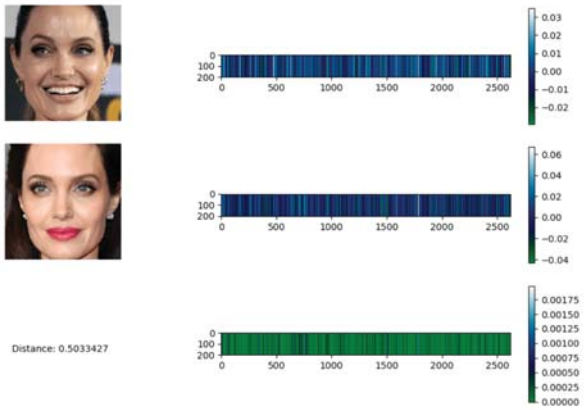

Fig. 2. Visualizing the calculation of Euclidean distance

Euclidean distance requires to find the squared difference of each dimension. Herein, we need to find the difference of each slot and find its squared value. The bar chart in the 3rd line stores these squared difference values. We finally need to sum of all slots and find its squared root value to find the Euclidean distance.

Cosine similarity of two vectors is derived by Euclidean dot product formula mentioned in Formula (3). Here, x and y are different multidimensional vectors. Similarity value is in range of [-1, +1]. Similar vectors should have a large similarity value whereas very different vectors should have a small similarity value. Vice versa, similar vectors should have a small value in Euclidean distance. That's why, we can convert it to cosine distance mentioned in Formula (4) which is 1 minus cosine similarity. In this way, smallness and largeness express same in the both cosine and Euclidean distances. Fig. 3 shows the cosine distance of some pairs on a

graph. Same person pairs have low cosine distance whereas different person pairs have high cosine distance.

$$cos\ \theta = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i{}^2}\sqrt{\sum_{i=0}^{n} y_i{}^2}} \quad (3)$$

$$D_C(x,y) = 1 - cos\theta \quad (4)$$

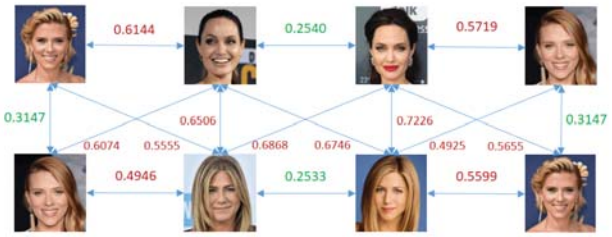

Fig. 3. Distance matrix for VGG-Face and cosine distance

We expect that images of same person should have a small distance whereas images of different people should have a large distance. We should define an optimum threshold value to determine an image pair is same person or not.

Threshold will be changed based on model and metric pair. The easiest way to determine an optimum threshold is to feed positive and negative instances to a decision tree algorithm. Because decision tree algorithms measure how well a feature separates the instances according to target classification. The best split points is found by C4.5 [31] algorithm in Table II. Besides, some accuracy scores are demonstrated for optimum threshold values.

TABLE II. BEST SPLIT POINT FOR THRESHOLD IN MODEL METRIC PAIRS

	Cosine	Euclidean	Euclidean L2
VGGFace	Threshold: 0.31 Accuracy: 89.28 Precision: 97.41 Recall: 80.71 F1: 88.28	Threshold: 0.47 Accuracy: 81.42 Precision: 97.82 Recall: 64.28 F1: 77.58	Threshold: 0.79 Accuracy: 89.28 Precision: 97.41 Recall: 80.71 F1: 88.28
FaceNet	Threshold: 0.40 Accuracy: 98.21 Precision: 100 Recall: 96.42 F1:98.18	Threshold: 11.26 Accuracy: 98.57 Precision: 100 Recall: 97.14 F1: 98.55	Threshold: 0.90 Accuracy: 98.21 Precision: 100 Recall: 96.42 F1: 98.18
OpenFace	Threshold: 0.11 Accuracy: 57.85 Precision: 95.83 Recall: 16.42 F1: 28.04	Threshold: 0.47 Accuracy: 57.85 Precision: 95.83 Recall: 16.42 F1: 28.04	Threshold: 0.47 Accuracy: 57.85 Precision: 95.83 Recall: 16.42 F1: 28.04
DeepFace	Threshold: 0.13 Accuracy: 54.64 Precision: 100 Recall: 9.28 F1: 16.99	Threshold: 42.21 Accuracy: 52.50 Precision: 100 Recall: 5.00 F1: 9.52	Threshold: 0.51 Accuracy: 54.64 Precision: 100 Recall: 9.28 F1: 16.99

Facebook researchers declared that human beings have 97.53% accuracy [13] in face recognition tasks for labeled faces in the wild data set [32]. Even though we've tested those pre-trained models on a custom data set, just single FaceNet model passed this level of accuracy.

In statistics, kernel density estimation (KDE) estimates the probability density function. This uses Gaussian kernels as well. So, Fig. 4 shows the distribution of each single model metric pair for yes and no classes. This distribution graphic could show the robustness of each model as well.

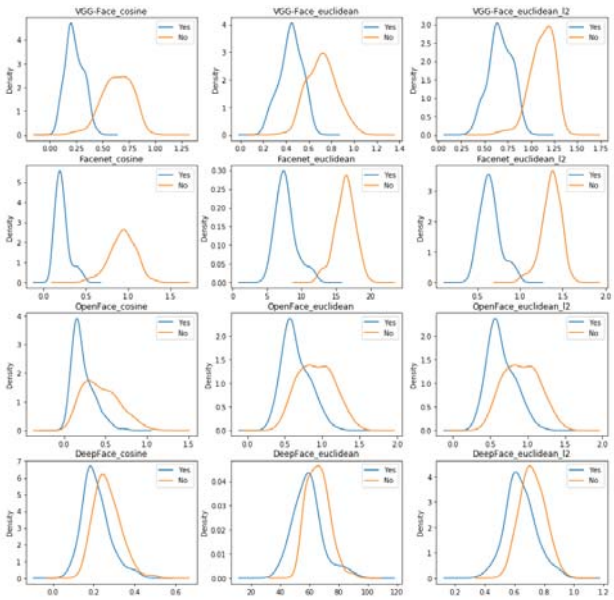

Fig. 4. Distance distributions

It seems that Facenet is the best single model supported in our framework and VGG-Face comes after it based on the accuracy results given in Table II. Besides, Euclidean L2 form seems to offer more robust than other distance metrics for each model.

III. SOFTWARE FRAMEWORK

We have mentioned all stages of a modern face recognition pipeline. Our framework handles of those stages in the background.

It builds a face recognition model in TensorFlow [8] and Keras [33]. Moreover, it downloads pre-trained weights of face recognition models externally because of the file sizes. Furthermore, it detects and aligns faces, represents image pairs as vectors based on passed face recognition model, finds distances between representations based on passed distance metric, and finally determines the pair is same person based on the optimum threshold. Everything is handled in the background.

The code sample mentioned in Table III will verify two image pairs are same person or not. Besides, its hybrid feature is shown. Face verification task could be handled by any state-of-the-art face recognition model. The output of the verification function is shown in Fig. 5.

TABLE III. FACE VERIFICATION ALGORITHM

```
#!pip install deepface
from deepface import DeepFace

models = ["VGG-Face", "Facenet", "OpenFace", "DeepFace",
"DeepID", "Dlib"]
metrics = ["cosine", "euclidean", "euclidean_l2"]

for model in models:
  for metric in metrics:
    obj = DeepFace.verify("img1.jpg", "img2.jpg"
                          , model_name = model
                          , distance_metric = metric)
    print(obj["verified"])
```

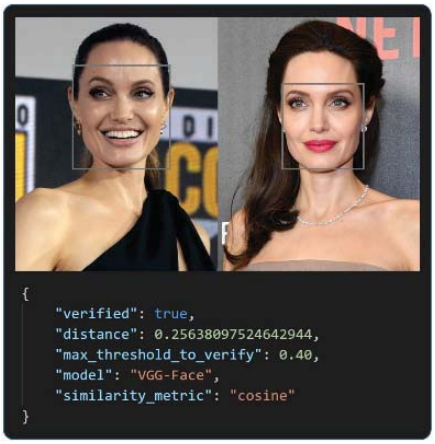

Fig. 5. The output of the verification function

Besides, finding a face in a large scale data set requires to apply face verification several times. The framework offers an out-of-the-box function for this task as well. The code block mentioned in Table IV will find the identity of a given image in the database. The output of the find function is demonstrated in Fig. 6.

TABLE IV. FACE RECOGNITION ALGORITHM

```
from deepface import DeepFace
import pandas as pd

models = ["VGG-Face", "Facenet", "OpenFace", "DeepFace",
"DeepID", "Dlib"]
metrics = ["cosine", "euclidean", "euclidean_l2"]

for model in models:
  for metric in metrics:
    df = DeepFace.find(img_path = "img1.jpg"
                       , db_path = "database"
                       , model_name = model
                       , distance_metric = metric)
    print(df.head())
```

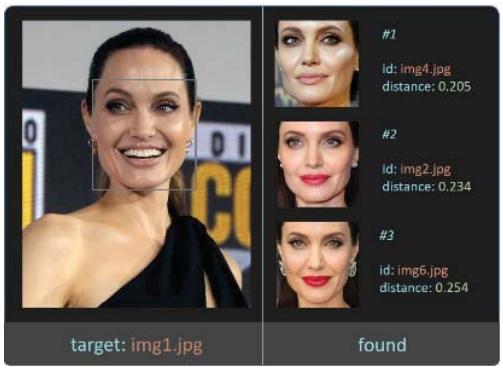

Fig. 6. The output of the find function

Representation is the costly operation in a face recognition task. However, we can extract and store representations beforehand. In this way, calling find function just requires to find embedding of the target image. We then find distances for all alternatives in the database but this could be completed very fast.

IV. CONCLUSION

In this paper, the requirements of a modern face recognition pipeline and focused on the stages of a pipeline to develop a framework from scratch was described. Then proposed lightweight software framework bridging a gap between machine learning studies and software engineering was explained. It wraps the most popular face recognition models. Its hybrid feature enables to switch the model among the state-of-the-art face recognition models. In this way, programmers could adapt human-level face recognition tasks with just a few lines of code.

As a future work, building an ensemble method covering all face recognition models and distance metrics to boost results was planned.

REFERENCES

[1] M. Kosugi, "Robust identification of human face using mosaic pattern and BPN," in *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, 1992, pp. 296–305, doi: 10.1109/NNSP.1992.253683.

[2] H. Kondo and S. B. A. Rahman, "Human-face recognition using neural network with mosaic pattern," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, 1999, vol. 6, pp. 831–834 vol.6, doi: 10.1109/ICSMC.1999.816659.

[3] Y. Ariki and W. Ishikawa, "Integration of face and speaker recognition by subspace method," in *Proceedings of 13th International Conference on Pattern Recognition*, 1996, vol. 3, pp. 456–460 vol.3, doi: 10.1109/ICPR.1996.546989.

[4] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, 1997, doi: 10.1109/34.598228.

[5] W. Zhao and R. B. T.-F. P. Chellappa, Eds., "INTRODUCTION TO THE STATISTICAL EVALUATION OF FACE-RECOGNITION ALGORITHMS," in *Face Processing*, Burlington: Elsevier, 2006, pp. 87–123.

[6] Z. Cheng, X. Zhu, and S. Gong, "Face re-identification challenge: Are face recognition models good enough?," *Pattern Recognit.*, vol. 107, p. 107422, 2020, doi: https://doi.org/10.1016/j.patcog.2020.107422.

[7] O. Yamaguchi, K. Fukui, and K.-. Maeda, "Face recognition using temporal image sequence," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 318–323, doi: 10.1109/AFGR.1998.670968.

[8] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016, pp. 265–283.

[9] Bhagirath, N. Mittal, and S. Kumar, "Machine Learning Computation on Multiple GPU's using CUDA and Message Passing Interface," in *2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC)*, 2019, pp. 18–22, doi: 10.1109/PEEIC47157.2019.8976714.

[10] H. Ando, Y. Niitsu, M. Hirasawa, H. Teduka, and M. Yajima, "Improvements of Classification Accuracy of Film Defects by Using GPU-accelerated Image Processing and Machine Learning Frameworks," in *2016 Nicograph International (NicoInt)*, 2016, pp. 83–87, doi: 10.1109/NicoInt.2016.15.

[11] I. Ghodgaonkar *et al.*, "Observing Responses to the COVID-19 Pandemic using Worldwide Network Cameras," *arXiv Prepr. arXiv2005.09091*, 2020.

[12] A. Goel *et al.*, "Observing Responses to the COVID-19 Pandemic using Worldwide Network Cameras," 2020.

[13] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708, doi: 10.1109/CVPR.2014.220.

[14] David Sandberg, "FaceNet," 2018. https://github.com/davidsandberg/facenet.

[15] Swarup Ghosh, "DeepFace," 2020. https://github.com/swghosh/DeepFace.

[16] Sefik Ilkin Serengil, "DeepFace." https://github.com/serengil/deepface.

[17] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," 2015, pp. 41.1-41.12, doi: 10.5244/c.29.41.

[18] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07-12-June, pp. 815–823, doi: 10.1109/CVPR.2015.7298682.

[19] T. Baltrusaitis, P. Robinson, and L. P. Morency, "OpenFace: An open source facial behavior analysis toolkit," in *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016, pp. 1–10, doi: 10.1109/WACV.2016.7477553.

[20] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1891–1898.

[21] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, 2014, pp. 1988–1996.

[22] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, 2009.

[23] A. Geitgey, "Face recognition," 2017. https://github.com/ageitgey.

[24] J. Sherrah and S. Gong, "Fusion of 2D face alignment and 3D head pose estimation for robust and real-time performance," in *Proceedings International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems. In Conjunction with ICCV'99 (Cat. No.PR00378)*, 1999, pp. 24–30, doi: 10.1109/RATFG.1999.799219.

[25] G. Bradski, "The OpenCV Library," *Dr. Dobb's J. Softw. Tools*, 2000.

[26] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[27] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07-12-June, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

[28] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[29] H. V Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6493 LNCS, no. PART 2, pp. 709–720, doi: 10.1007/978-3-642-19309-5_55.

[30] M. D. Malkauthekar, "Analysis of Euclidean distance and Manhattan distance measure in face recognition," in *IET Conference Publications*, 2013, vol. 2013, no. CP646, pp. 503–507, doi: 10.1049/cp.2013.2636.

[31] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *J. Artif. Intell. Res.*, vol. 4, pp. 77–90, 1996, doi: 10.1613/jair.279.

[32] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments." 2007.

[33] F. Chollet and others, "Keras." 2015.