A Benchmark of Facial Recognition Pipelines and Co-Usability Performances of Modules

Araştırma Makalesi/Research Article

Şefik İlkin SERENGİL [1], Alper ÖZPINAR [2]

[1] Solution Engineering Department, Vorboss Limited, London, UK
[2] Mechatronics Engineering Department, Istanbul Ticaret University, Istanbul, Turkey
sefik.serengil@vorboss.com, alper.ozpinar@ticaret.edu.tr

(Geliş/Received:01.12.2023; Kabul/Accepted:29.03.2024)
DOI: 10.17671/gazibtd.1399077

Abstract**—** Researchers from leading technology companies, prestigious universities worldwide, and the open-source community have made substantial strides in the field of facial recognition studies in recent years. Experiments indicate that facial recognition approaches have not only achieved but surpassed human-level accuracy. A contemporary facial recognition process comprises four key stages: detection, alignment, representation, and verification. Presently, the focus of facial recognition research predominantly centers on the representation stage within the pipelines. This study conducted experiments exploring alternative combinations of nine state-of-the-art facial recognition models, six cutting-edge face detectors, three distance metrics, and two alignment modes. The co-usability performances of implementing and adapting these modules were assessed to precisely gauge the impact of each module on the pipeline. Theoretical and practical findings from the study aim to provide optimal configuration sets for facial recognition pipelines.

Keywords**—** facial recognition, face verification, deepface, vector models

Yüz Tanıma Üretim Hatlarının ve Modüllerinin Birlikte Kullanımının Karşılaştırılması

Özet**—** Son yıllarda, büyük teknoloji şirketleri, dünyanın önde gelen üniversiteleri ve açık kaynak topluluğundan gelen araştırmacılar, yüz tanıma alanında önemli ilerlemeler kaydetmişlerdir. Yapılan deneyler, yüz tanıma yaklaşımlarının insan düzeyinde doğruluk sağladığını ve hatta aştığını göstermektedir. Modern bir yüz tanıma süreci genellikle dört aşamadan oluşur: algılama, hizalama, temsil ve doğrulama. Mevcut yüz tanıma çalışmaları genellikle üretim hatlarındaki temsil aşamasına odaklanmıştır. Bu çalışmada, dokuz farklı son teknoloji yüz tanıma modeli, altı son teknoloji yüz dedektörü, üç mesafe ölçümü ve iki hizalama modunun farklı kombinasyonları için çeşitli deneyler gerçekleştirilmiştir. Bu modüllerin uygulanması ve uyumlu hale getirilmesinin genel performansları, her bir modülün üretim hattındaki spesifik etkisini belirlemek amacıyla değerlendirilmiştir. Çalışmanın teorik ve pratik sonucu olarak, yüz tanıma hatları için en iyi konfigürasyon setlerinin paylaşılması hedeflenmektedir.

Anahtar Kelimeler**—** yüz tanıma, yüz doğrulama, deepface, vektör modeller

1. INTRODUCTION

In recent years, researchers and open-source communities have made substantial advancements in the field of facial recognition. This area of research has garnered attention from major technology companies and top universities worldwide. Experiments indicate that machine learning-based facial recognition approaches developed by these entities have not only achieved but surpassed human-level accuracy.

A widely accepted perspective among facial recognition researchers advocates for the incorporation of four fundamental stages in contemporary facial recognition pipelines [1]: detection, alignment, representation, and verification. Notably, the representation stage stands out as the most crucial phase, tasked with generating vector embeddings from facial images. Presently, state-of-the-art models for the representation module predominantly leverage convolutional neural networks, exemplified by models like FaceNet [2], VGG-Face [3], ArcFace [4], Dlib [5], Sface [6], OpenFace [7], DeepFace [1], and DeepId [8].

Following the representation stage, the verification module calculates distances between vector embeddings of face pairs. Experiments reveal that the distribution of distances for pairs belonging to the same person and different individuals is discrete, particularly in robust models [9]. Various metrics, such as Euclidean, L2 normalized Euclidean form, or cosine similarity, can be employed to compute the distances between vectors.

Prior to the representation stage, both detection and alignment serve as early modules in the pipeline, aiming to provide clear inputs. Substantial progress has been made in recent years in the field of face detection. Noteworthy face detectors include OpenCV Haar Cascade [10], SSD [11], MediaPipe [12] [13], MTCNN [14], Dlib HOG [5], and RetinaFace [15]. Some of these detectors are built on modern convolutional neural networks, such as MTCNN, RetinaFace, and SSD, while others adhere to legacy approaches like OpenCV and Dlib HOG. Additionally, facial landmarks, including the coordinates of the eyes, are provided by these detectors, facilitating the horizontal alignment of faces.

In the realm of facial recognition studies, there is, regrettably, a lack of uniformity as researchers and model creators approach the problem from diverse perspectives. The primary emphasis tends to be on the representation stage within the pipeline, where various combinations of face detectors, alignment procedures, and distance metrics are employed. This variability introduces a challenge in fairly comparing these models, as a facial recognition model may demonstrate superior performance when configured differently, potentially exceeding its initially reported score. The precise influence of detection, alignment, and verification modules on the overall pipeline remains to be fully understood.

This study assesses the efficacy of facial recognition pipelines across four key dimensions: facial recognition models, face detectors, distance metrics, and the impact of enabling or disabling alignment modes. The objective is to identify optimal configuration settings for the pipeline. The paper further seeks to elucidate the tangible influence of each module on the overall pipeline score. Ultimately, the study compares the performance of human subjects to that of different models and configurations, determining which ones surpassed, matched, approached, or fell short of human-level accuracy.

In this context, LightFace [9] [16] emerges as one of the most widely used facial recognition libraries for the Python programming language. Comprehensive in its coverage, it includes various facial recognition models, face detectors, distance metrics, and alignment modes, seamlessly managing all facial recognition procedures in the background. For the purposes of this study, the LightFace package was utilized and implemented to execute a range of experimental setups.

2. DATASET

The Labeled Faces in the Wild (LFW) dataset [17] stands as the predominant dataset in facial recognition studies, comprising 13,233 images belonging to 5,749 identities. For the purpose of this study, the LFW dataset was loaded using scikit-learn [18]. The dataset includes 2,200 pairs for training, 6,000 pairs for 10-fold cross-validation, and 1,000 pairs for testing. Each pair is appropriately labeled as either depicting the same person or different individuals. The original size of images in LFW is (250, 250, 3). Given the focus on pre-trained models in this study, attention is directed solely to the test set.

Half of the pairs of recommended test sets belong to the same person, whereas another half of the pairs correspond to different persons. So, the test set has fairly homogeneous target labels. That is why accuracy scores are mentioned in the performance results. The raw unaligned LFW images were used because detection and alignment impact in the pipeline were evaluated.

There are successful studies in the literature with good results in capturing information from the context such as image background or hairstyle. Neeraj Kumar et al [19] reported the performance of human-beings on LFW dataset. They collected the classification votes of 10 voters for each pair in LFW dataset consisting of 6000 instances via Amazon Mechanical Turk. Besides, voters taken three different tests: verifying original image pairs, tight crop images and inverse crops as shown in Figure 1.

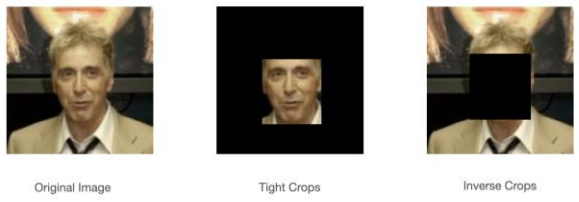

Figure 1. Test Tasks

Humans are able to verify uncropped face pairs with 99.2% accuracy score. They also obtain 94.2% accuracy even for image pairs on same dataset with blurred or blacked facial areas. On the other hand, human-beings have 97.5% accuracy for verifying cropped face pairs. So, people can find out a lot information from the context such as background. Regions outside of the facial area is not strongly recommended to be used not to boost results artificially. That is why, facial recognition researchers always compare built model performances with this score.

3. PIPELINE

A modern facial recognition pipeline consists of 4 common stages: detection, alignment, representation and verification. This section outlines the current procedures in place for these modules.

3.1. Detection and Alignment

Only open source face detectors were used in the detection module. There are five of them: OpenCv, Ssd, MtCnn, Dlib, and RetinaFace. Their role will be to detect faces and discard regions outside of the facial area in order to avoid artificially boosting results.

An alignment step is also performed after detection. The eye locations can then be used to construct a right-angled triangle with two corners at eye locations as shown in Figure 2.

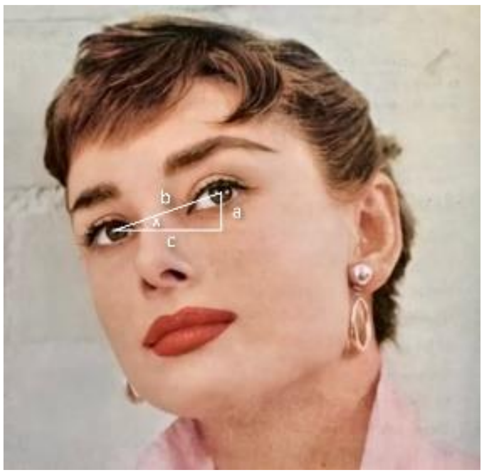

Figure 2. Alignment Procedure

Calculating the cosine of an angle is then done using the cosine rule in Formula 1. Inverse cosine of this term returns angle in radian. The angle can then be converted to degrees using the formula 2. The image will then be rotated by A degrees until the eyes are horizontal.

$$\cos(A) = \frac{b^2+c^2-a^2}{2bc} \tag{1}$$

$$A^{\circ} = \arccos\left(\frac{b^2+c^2-a^2}{2bc}\right)\frac{180}{\pi} \tag{2}$$

3.2. Resizing

The outcomes of the detection and alignment modules serve as inputs for the representation module. In facial recognition pipelines, the representation module predominantly relies on convolutional neural networks (CNNs). CNNs typically require inputs to have a fixed shape, yet detected and aligned facial images lack such uniformity. Table 1 provides an overview of the anticipated architectures for the facial recognition models discussed in this study including input shape, embedding dimensions, total parameters (weights and biases) and number of layers. Consequently, resizing the detected and aligned faces becomes imperative before invoking the representation module.

Table 1. Facial Recognition Model's Architectures

Model	Input Shape	Embedding Dimensions	Total Params	Num of Layers
FaceNet-128d	160, 160, 3	128	22M	447
FaceNet-512d	160, 160, 3	512	23M	447
Vgg-Face	224, 224, 3	4096	134M	36
ArcFace	112, 112, 3	512	34M	162
Dlib	150, 150, 3	128	63M	34
SFace	112, 112, 3	128	9M	88
OpenFace	96, 96, 3	128	3M	166
DeepFace	152, 152, 3	4096	102M	9
DeepId	55, 47, 3	160	395K	17

In order to prevent deformation, black padding pixels were introduced to the images instead of resizing them. Figure 3 visually depicts the outcomes of detection and alignment using various detectors, along with the addition of black padding pixels to resize the image to the expected input shape of VGG-Face.

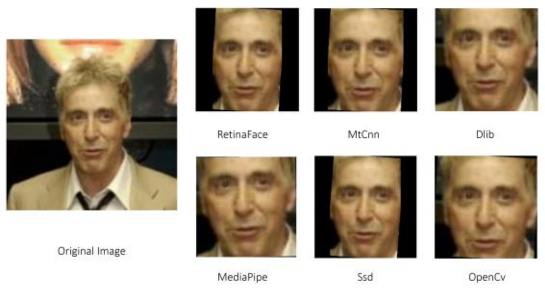

Figure 3. Detection, Alignment and Resizing

3.3. Representation

The representation module of all the facial recognition models discussed in this study is based on convolutional neural networks. In spite of the fact that CNN structures have many output nodes, units were not used for classification tasks. Rather than finding the most dominant output node to make classification, the probabilities of output nodes were used. Figure 4 shows how DeepFace [1] model finding vector embeddings.

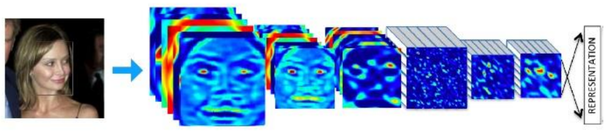

Figure 4. Vector Representation with DeepFace [1]

3.4. Verification

The representation module returns vectors for face pairs, and these vectors are passed to the verification module. The distance between those vectors was calculated in the verification module. Formula 3 shows the euclidean distance formula where p and q are n-dimensional vectors. Similarly, Formula 4 finds the cosine distance where p and q are n-dimensional vectors.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \qquad (3)$$

$$d(p,q) = 1 - \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} {p_i}^2}\sqrt{\sum_{i=1}^{n} {q_i}^2}} \qquad (4)$$

Before feeding the vectors to distance formulas, l_2 were applied for normalization to vectors to have smoother ones as shown in Formula 5 where q is a n-dimensional vector.

$$l_2(q) = \frac{q}{\sqrt{\sum_{i=1}^{n} {q_i}^2}} \qquad (5)$$

Accordingly, this pair would be classified as the same person if the distance was less than the threshold. Similarly, this pair could be classified as different persons if the distance between them exceeds the threshold value.

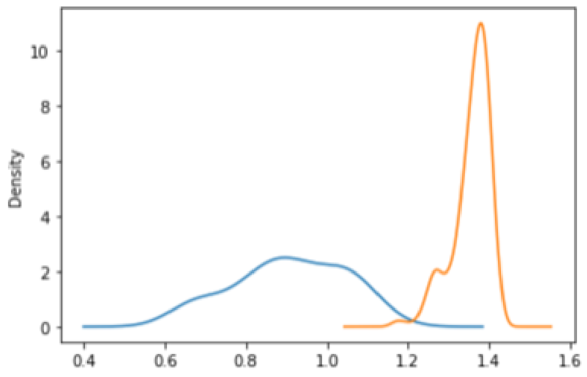

Figure 5. Distance Distribution

The experiments demonstrate that the distance distributions for the same person and different persons classes, as tested with the VGG-Face facial recognition model using L2 Normalized Euclidean distance and enabled alignment mode on unit test items of DeepFace, can be separated, as illustrated in Figure 5. In this context, the C4.5 algorithm [20] was employed to identify the threshold that yields the maximum information gain.

3.4. Handling Many Faces Issue

Some instances of the LFW dataset have some problematic data. For instance, an image may contain many faces and the one targeted by the label may not be known precisely. In this case, all pairs of faces were extracted using the current running detector (e.g. MtCnn), and all candidates were then represented as vectors using the current running facial recognition model (e.g. VGG-Face). Following that, all vector candidates were distanced by using current running similarity algorithms (e.g. cosine and euclidean). As a result, the face pair with the minimum distance was fed into the algorithm. Figure 6 explains how we extract the right face from these problematic samples. The faces of the lady are used in both images for this case since the distance is shorter.

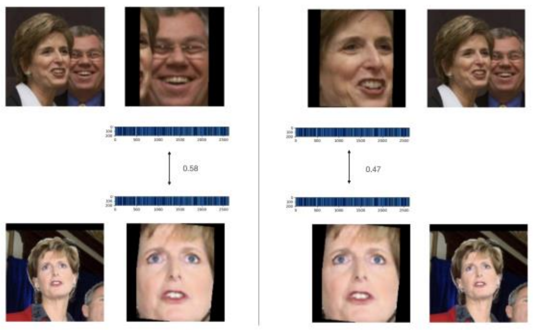

Figure 6. Picking a Face Among Many Faces

4. PRE-TRAINED MODELS

LightFace for python [21] will be used to build facial recognition pipelines. This library basically wraps the following models with pre-trained weights.

4.1. Facial Recognition Models

Researchers from the Visual Geometry Group at Oxford University developed the VGG-Face model. The model's structure was detailed in their paper [3], and they generously provided pre-trained weights for the VGG-Face model in MatLab, Torch, and Caffe on the research group's website [22]. Subsequently, the MatLab weights were converted into Keras, and the VGG-Face model was reconstructed from the ground up using Keras.

Researchers at Google designed the FaceNet model, which outputs 128 dimensions. The model's structure is outlined in their paper [2], but the pre-trained weights are not publicly available. However, David Sandberg undertook the task of re-training FaceNet with both 128 and 512 dimensional outputs and generously shared the pre-trained weights in TensorFlow format [23]. These TensorFlow weights were then converted to Keras format, and a new model in Keras was constructed from the ground up.

ArcFace, a component of the InsightFace project, was developed by the Deep Inside research group [4]. The ArcFace model, along with its pre-trained weights, can be accessed on the group's website for both PyTorch and MXNet. Additionally, Leon D. Garse re-trained the ArcFace model specifically for Keras [24]. The pre-trained weights from this repository were utilized, and a new model was constructed from scratch in Keras.

SFace, primarily developed by Yaoyao Zhong [6], includes shared models and pre-trained weights accessible for both PyTorch and MXNet. The OpenCV community played a role in converting the model to a language-independent ONNX format [25]. Consequently, SFace can be utilized through OpenCV in this context.

Researchers from Carnegie Mellon University developed the OpenFace model. Both the model and its pre-trained weights can be accessed on the research group's website for Lua Torch. Victor Sy Wang took on the task of converting the pre-trained weights of the OpenFace model to Keras [26]. The weights from this re-implementation were utilized, and the model was then constructed from scratch.

Facebook researchers developed the DeepFace model, outlining its structure in their paper [1], although they did not provide the pre-trained weights for the model. Swarup Ghosh took the initiative to re-train the DeepFace model using Keras from scratch and shared the resulting pre-trained weights in his repository [27]. While the original study utilized the Social Face Classification (SFC) [1] dataset with 4030 identities, Ghosh re-trained the DeepFace model using the VGGFace2 [28] dataset with 8631 identities. Consequently, the original DeepFace model produces 4030-dimensional vectors, whereas Ghosh's re-implementation generates 8631-dimensional vectors. The DeepFace model was reconstructed from the ground up, and the pre-trained weights from this re-implementation were utilized.

The DeepId model, crafted by researchers from the Chinese University of Hong Kong, had its model structure detailed in their paper [8]. However, the pre-trained weights were not made publicly available. Roy Ran took on the task of re-training the DeepId model from the ground up and generously shared the resulting pre-trained weights specifically for TensorFlow [29]. To facilitate utilization of these pre-trained weights, a conversion process was employed to adapt them into the Keras format.

Dlib constructs a ResNet model designed for facial recognition purposes [5], with the Dlib library being directly imported for implementation.

4.3. Face Detectors

The MtCnn face detector model was developed by Kaipeng Zhang et al [14]. Iván de Paz Centeno took the initiative to independently re-implement MtCnn from scratch using Keras, presenting it as a project [30]. The MtCnn project has been directly incorporated into the system.

The RetinaFace model, created by researchers from Deep Insight as part of the InsightFace project [4], is provided with both the model and pre-trained weights for PyTorch and MXNet on the research group's website. Additionally, Stanislas Bertrand undertook the task of re-implementing the RetinaFace model using Keras [31]. Following the completion of the RetinaFace project repackaging [32], this new RetinaFace package was directly imported for use.

The SSD (Single Shot Multibox Detector) model, designed by Wei Liu et al [11], had its Caffe model wrapped by the OpenCV community [10]. In practice, SSD was utilized through OpenCV. However, SSD itself does not provide facial landmarks. To align faces identified by SSD, the eye detection module of OpenCV was employed. MediaPipe [12] features a face detection module built on BlazeFace [13], which is structurally derived from SSD but includes facial landmarks. The MediaPipe library was directly imported for use. Additionally, Dlib was employed for face detection using its HOG model [5], and the Haar Cascade module of OpenCV was imported to utilize its face detection functionality.

Re-implementations of certain reference models have been undertaken, and as a consequence, experimental results will be linked to these re-implementations rather than the original studies. It's important to note that even when the same facial recognition model and pre-trained weights were utilized, discrepancies in pre-processing techniques and model implementation may exist compared to those employed in the original study. Specifically, a 2D discarding approach was implemented, discarding regions outside the facial area using commonly available open-source face detectors, as suggested in the literature [19], to prevent the artificial inflation of results.

4. EXPERIMENTS

A total of 378 experiments were performed on the LFW dataset test set, considering combinations of the four dimensions listed in Table 8. This exhaustive approach aimed to identify the optimal configuration set for facial recognition. Additionally, each module in the pipeline was systematically examined to discern its specific impact on the overall results.

Table 8. Different Configration Sets for Experiments

Configuration	Possible Values
Facial Recognition Models	FaceNet128d, FaceNet512d, Vgg-Face, ArcFace, Dlib, SFace, OpenFace, DeepFace, DeepId
Face Detector Models	OpenCv, Ssd, MtCnn, Dlib, RetinaFace, MediaPipe, No Detection
Distance Metrics	Cosine, Euclidean, L2-Norm Euclidean
Alignment Mode	Enabled, Disabled

Tables 2, 3, 4, 5, 6, and 7 present the performance matrices of facial recognition models and face detectors across various distance metrics and alignment modes. In these tables, columns denote facial recognition models, rows represent face detectors, and the cells display corresponding accuracy values. Additionally, to enhance comprehension of the influence of detection and alignment modules, the scores of facial recognition models without detection and alignment are also provided.

As outlined in Section 3, the output of the representation module consists of a vector pair, and the subsequent step involves measuring the distance between these vectors. A pair is classified as the same person if the distance between them falls below the specified threshold value. Furthermore, ROC curves were generated using distance values and corresponding target labels. This approach is a conventional practice to assess the model's performance from a holistic perspective.

ROC curves for different models were illustrated in Figure 7 for FaceNet512d, Figure 8 for VGG-Face, Figure 9 for Dlib, Figure 10 for FaceNet128d, Figure 11 for ArcFace, and Figure 12 for SFace, focusing specifically on robust detectors (RetinaFace, MtCnn, Dlib, and MediaPipe). It is recommended to view these figures in color for optimal clarity. Additionally, the graphics include area under the curve (AUC) values, and the legends are arranged in descending order based on these values. In this context, a higher AUC value indicates superior model performance.

The ROC curves presented above highlight that FaceNet512d and VGG-Face stand out as the most robust models, regardless of the detector used. However, the performance of other facial recognition models exhibits variations depending on the specific configuration employed.

5. RESULTS

The experiments examined both context-independent and context-dependent conditions. While there is significant importance in running facial recognition pipelines in a context-independent manner, the results obtained under context-dependent conditions may provide a more accurate understanding of the direct influence of face detection on facial recognition pipelines. The study involved comparing these models with human performance, resulting in a context-free accuracy of 97.5% and a non-context-free accuracy of 99.2% [19]. Finally, the study demonstrated the impact of detectors and alignment modules on the pipelines.

Table 9. Comparison of Measured Accuracy Scores in LightFace and Declared Accuracy Scores in Their Original Researches

Model	Measured Accuracy	Declared Accuracy	Original Paper
FaceNet-512d	98.4	99.6	[2]
FaceNet-128d	97.0	99.6	[2]
Dlib	96.8	99.3	[5]
VGG-Face	96.7	98.9	[3]
ArcFace	96.6	99.5	[4]
SFace	93.0	99.5	[6]
OpenFace	78.7	92.9	[7]
DeepFace	68.7	97.3	[1]
DeepId	65.6	97.4	[8]

As shown in Table 9, the measured accuracies of various facial recognition models using the LightFace library offer valuable insights into their performance. A comparison with the accuracy scores declared in their original papers reveals that FaceNet, Dlib, ArcFace, and VGG-Face closely approach the declared performances, albeit falling slightly below. Notably, for models such as VGG-Face and Dlib, the small differences may be a result of the adopted normalization techniques in both the original studies and the LightFace framework. Additionally, it is noteworthy that LightFace employs open-sourced pre-trained weights for their re-implementation instead of using the original ones, as seen in the case of FaceNet and ArcFace. These minor differences are deemed acceptable and do not compromise the overall robustness of these models. Despite the slight variations, the models remain robust and showcase reliable performance. Despite these differences, SFace, while exhibiting a performance lower than declared, shows a manageable gap, suggesting its continued usability.

In contrast, OpenFace, DeepFace, and DeepId exhibit notable disparities from their originally reported accuracies, which can be attributed to the utilization of pre-trained weights retrieved from open-source re-implementations rather than the original studies' pre-trained weights. The original pre-trained weights, as detailed in Section 4, are not publicly available. This distinction highlights the significance of the source of pre-trained weights, as the use of open-source alternatives may lead to differences in model performance. The observed disparities underscore the importance of obtaining and utilizing the original pre-trained weights when aiming to replicate or surpass the accuracies reported in the respective original studies.

5.1. Context Independent Results

FaceNet512d achieved an accuracy of 98.4%, followed by FaceNet128d with 97.0% accuracy. Both models have surpassed human-level accuracy. The most satisfactory configurations are highlighted in bold within the tables.

Dlib achieved an accuracy of 96.8%, ArcFace and VGG-Face demonstrated 96.7% and 96.6% accuracy, respectively. These models achieved accuracy levels comparable to human performance. SFace, with an accuracy of 93.0%, approached human-level accuracy.

OpenFace achieved an accuracy of 78.7%, DeepFace reached 68.7%, and DeepId attained 65.6% accuracy. Consequently, these re-implementations exhibited subpar performance, rendering them unsuitable for deployment in production pipelines. Notably, FaceNet128d, ArcFace, and Dlib displayed significantly reduced performance when the detection stage was omitted. It is imperative to execute the detection stage for these facial recognition models.

OpenCv and Ssd detectors exhibit diminished performance when the alignment mode is enabled. This limitation arises from their inability to pinpoint eye locations with the same precision as RetinaFace or MtCnn. Consequently, it is advisable to omit the alignment stage when using these detectors. RetinaFace, MtCnn, and Dlib, on the other hand, showcase a notable performance enhancement when the alignment mode is activated, establishing them as robust detectors across all considered configurations. The performance of MediaPipe experiences a slight decline with alignment; even with alignment disabled, it ranks below RetinaFace, Dlib, and MtCnn.

Finally, none of the distance metrics demonstrate underperformance, and despite their variances, each contributes significantly to the overall performance.

5.2. Context Dependent Results

In the presence of contextual information, facial recognition models struggle to achieve human-level accuracy. Their performance sees a significant boost when focusing exclusively on the facial region, likely due to their training on context-independent data. FaceNet512d attains a 92% score even with detection disabled, marking the closest performance to human among facial recognition models without detectors. VGG-Face achieves a 90.6% score, while SFace scores 83.4% with detection turned off, both trailing FaceNet-512d. FaceNet512d stands out as the optimal choice for use cases demanding both swift and reliable results, such as facial recognition in crowded environments. In such scenarios, detection and alignment may be skipped, yet the pipeline maintains a commendable score.

5.3. Impact of custom modules in the pipeline

In optimal conditions, the detection impact for ArcFace is 41.8%, FaceNet128d is 30.8%, Dlib is 27.3%, OpenFace is 20.9%, SFace is 9.6%, DeepFace is 7.6%, FaceNet-512d is 6.6%, VGG-Face is 6.1%, and DeepId is 3.6%. Therefore, the inclusion of a detection module is essential for facial recognition pipelines.

Similarly, the alignment impact for SFace is 17.2%, DeepFace is 13.7%, ArcFace is 11.7%, FaceNet128d is 6.5%, VGG-Face is 1.6%, OpenFace is 1.0%, FaceNet-512d is 0.7%, and DeepId is 0.7% in the best-case scenarios. Consequently, alignment serves as a crucial enhancement module for facial recognition pipelines.

Furthermore, identifying the intersections of misclassifications for each facial recognition model across various configuration sets will aid in comprehending the weaknesses inherent in each model. As depicted in Figure 13, FaceNet512d misclassifies 6 pairs, FaceNet128d misclassifies 5 pairs, and both Dlib and ArcFace misclassify 3 pairs in all experiments involving Euclidean, l2 normalized Euclidean, and Cosine distance metrics, as well as RetinaFace, MtCnn, and Dlib face detectors, with alignment mode exclusively enabled.

Predominantly, the common misclassifications for each model involve false negatives. It appears that these models commonly misclassify pairs when individuals wear accessories like sunglasses or caps. Similarly, differences in age and emotion within pairs contribute to misclassifications. Furthermore, the common misclassifications mainly pertain to pairs of the same person.

6. CONCLUSION

In this study, all configuration alternatives were tested in LightFace. To avoid artificially boosting results, the facial area in images was focused and the regions outside of the facial area were discarded.

Based on the results of the experiments, it was determined that some facial recognition models were capable of reaching or exceeding human level accuracy under certain configurations. It can be concluded that the facial recognition model with dependent face detector, distance metric and alignment mode underperform or overperform. Based on their best configuration set, facial recognition models were classified into four categories: passing human level accuracy, reaching human level accuracy, coming close to human level accuracy, and underperforming ones. AUC and ROC curves also provide opinions about the robustness of facial recognition models.

Additionally, this work discusses the contribution of detection and alignment modules to pipeline accuracy. The detection module is an essential component of a pipeline as it can improve performance by up to 40%. While alignment serves as a significant performance booster, improving performance by up to 17%. Results found regarding the robustness of face detectors. OpenCv, SSD and MediaPipe have deteriorated when alignment is enabled, whereas RetinaFace, MtCnn and Dlib are always resilient.

LightFace also features an easy and simple interface. The dimensions mentioned in this study are merely input arguments for a pipeline. Thus, practitioners are able to construct and run facial recognition pipelines with a few lines of code. This study can be used as a guide to assist practitioners in choosing the most appropriate configurations for their facial recognition pipelines according to their individual needs.

Table 2. Accuracy Metric for Euclidean Distance and Disabled Alignment

	FaceNet-128d	*FaceNet-512d*	*VGG-Face*	*ArcFace*	*Dlib*	*SFace*	*OpenFace*	*DeepFace*	*DeepId*
RetinaFace	92.8	96.1	95.7	84.1	88.3	78.6	70.8	67.4	64.3
MtCnn	92.5	95.9	95.5	81.8	89.3	76.3	70.9	65.9	63.2
Dlib	89.0	96.0	94.1	82.6	96.3	73.1	75.9	61.8	61.9
MediaPipe	87.1	94.9	93.1	71.1	91.9	73.2	77.6	61.7	62.4
Ssd	94.9	97.2	96.7	83.9	88.6	82.0	69.9	66.7	64.0
OpenCv	90.2	94.1	95.8	89.8	91.2	86.9	71.1	68.4	61.1
None	64.1	92.0	90.6	56.6	69.0	81.4	57.4	60.8	60.7

Table 3. Accuracy Metric for L2 Normalized Euclidean Distance and Disabled Alignment

	FaceNet-128d	*FaceNet512d*	*VGG-Face*	*ArcFace*	*Dlib*	*SFace*	*OpenFace*	*DeepFace*	*DeepId*
RetinaFace	95.9	**98.0**	95.7	95.7	88.4	90.6	70.8	67.7	64.6
MtCnn	96.2	**97.8**	95.5	95.9	89.2	91.1	70.9	67.0	64.0
Dlib	89.9	96.5	94.1	93.8	95.6	75.0	75.9	62.6	61.8
MediaPipe	90.0	96.3	93.1	89.3	91.8	74.6	77.6	64.9	61.6
Ssd	97.0	**97.9**	96.7	96.6	89.4	93.0	69.9	68.7	64.9
OpenCv	92.9	96.2	95.8	93.2	91.5	91.7	71.1	68.3	61.6
None	67.6	91.4	90.6	57.2	69.3	83.4	57.4	62.6	61.6

Table 4. Accuracy Metric for Cosine Distance and Disabled Alignment

	FaceNet-128d	*FaceNet512d*	*VGG-Face*	*ArcFace*	*Dlib*	*SFace*	*OpenFace*	*DeepFace*	*DeepId*
RetinaFace	95.9	**98.0**	95.7	95.7	88.4	90.6	70.8	67.7	63.7
MtCnn	96.2	**97.8**	95.5	95.9	89.2	91.1	70.9	67.0	64.0
Dlib	89.9	96.5	94.1	93.8	95.6	75.0	75.9	62.6	61.7
MediaPipe	90.0	96.3	93.1	89.3	91.8	74.6	77.6	64.9	61.6
Ssd	97.0	**97.9**	96.7	96.6	89.4	93.0	69.9	68.7	63.8
OpenCv	92.9	96.2	95.8	93.2	91.5	91.7	71.1	68.1	61.1
None	67.6	91.4	90.6	54.8	69.3	83.4	57.4	62.6	61.1

Table 5. Accuracy Metric for Euclidean Distance and Enabled Alignment

	FaceNet-128d	*FaceNet-512d*	*VGG-Face*	*ArcFace*	*Dlib*	*SFace*	*OpenFace*	*DeepFace*	*DeepId*
RetinaFace	93.5	95.9	95.8	85.2	88.9	80.2	69.4	67.0	65.6
MtCnn	93.8	95.2	95.9	83.7	89.4	77.4	70.2	66.5	63.3
Dlib	90.8	96.0	94.5	88.6	96.8	66.3	75.8	63.4	60.4
MediaPipe	88.6	95.1	92.9	73.2	93.1	72.5	78.7	61.8	62.2
Ssd	85.6	88.9	87.0	75.8	83.1	76.9	66.8	63.4	62.5
OpenCv	84.2	88.2	87.3	73.0	84.4	81.1	66.4	65.5	59.6
None	64.5	92.0	90.6	56.6	69.0	81.4	57.4	60.8	60.7

Table 6. Accuracy Metric for L2 Normalized Euclidean Distance and Enabled Alignment

	FaceNet-128d	*FaceNet-512d*	*VGG-Face*	*ArcFace*	*Dlib*	*SFace*	*OpenFace*	*DeepFace*	*DeepId*
RetinaFace	96.4	**98.4**	95.8	96.6	89.1	92.4	69.4	67.7	64.4
MtCnn	96.8	**97.6**	95.9	96.0	90.0	90.5	70.2	66.4	64.0
Dlib	92.6	97.0	94.5	95.1	96.4	69.8	75.8	66.5	59.5
MediaPipe	90.6	96.1	92.9	90.3	92.6	75.4	78.7	64.7	63.0
Ssd	87.5	88.7	87.0	86.2	83.3	84.6	66.8	64.1	62.6
OpenCv	84.8	87.6	87.3	84.6	84.0	83.6	66.4	63.8	60.9
None	67.6	91.4	90.6	57.2	69.3	83.4	57.4	62.6	61.6

Table 7. Accuracy Metric for Cosine Distance and Enabled Alignment

	FaceNet128	*FaceNet-512d*	*VGG-Face*	*ArcFace*	*Dlib*	*SFace*	*OpenFace*	*DeepFace*	*DeepId*
RetinaFace	96.4	**98.4**	95.8	96.6	89.1	92.4	69.4	67.7	64.4
MtCnn	96.8	**97.6**	95.9	96.0	90.0	90.5	70.2	66.3	63.0
Dlib	92.6	97.0	94.5	95.1	96.4	69.8	75.8	66.5	58.7
MediaPipe	90.6	96.1	92.9	90.3	92.6	75.4	78.7	64.8	63.0
Ssd	87.5	88.7	87.0	86.2	83.3	84.5	66.8	63.8	62.6
OpenCv	84.9	87.6	87.2	84.6	84.0	83.6	66.2	63.7	60.1
None	67.6	91.4	90.6	54.8	69.3	83.4	57.4	62.6	61.1

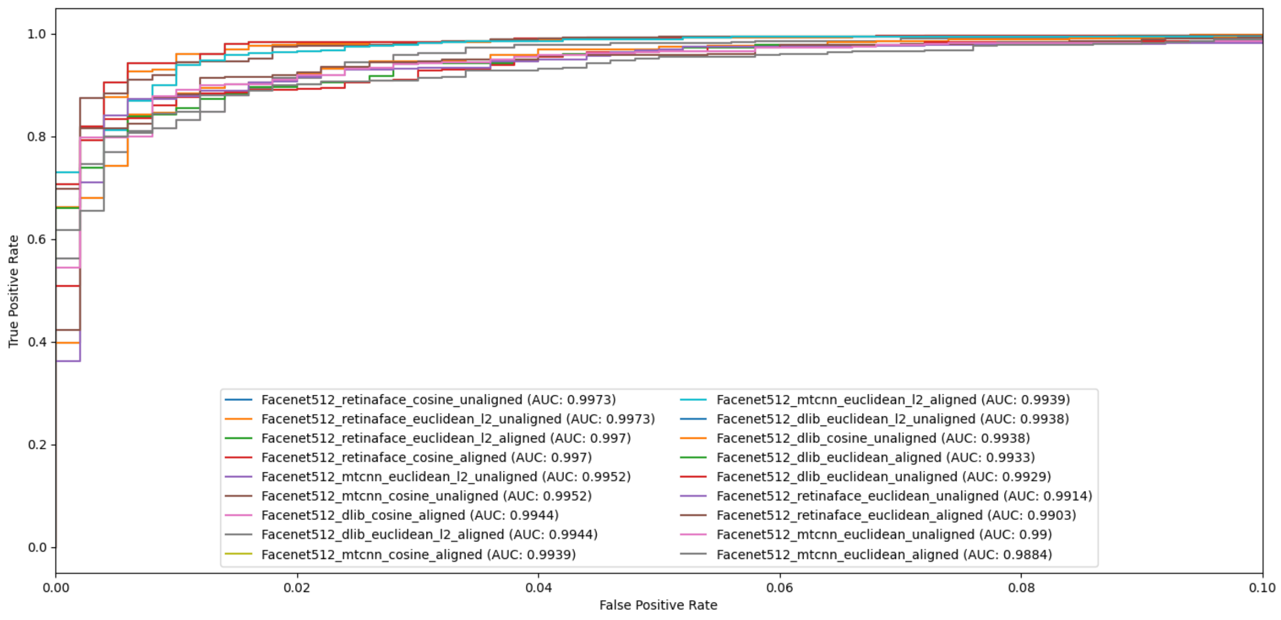

Figure 7. ROC Curves and AUC Scores for FaceNet512d

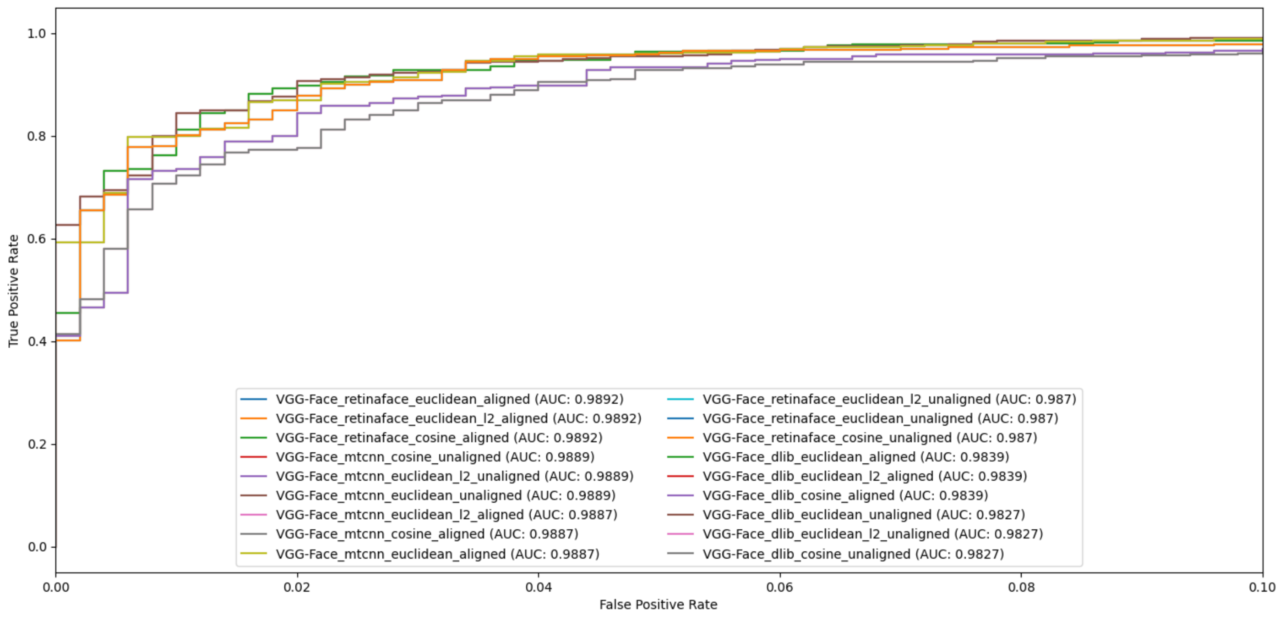

Figure 8. ROC Curves and AUC Scores for VGG-Face

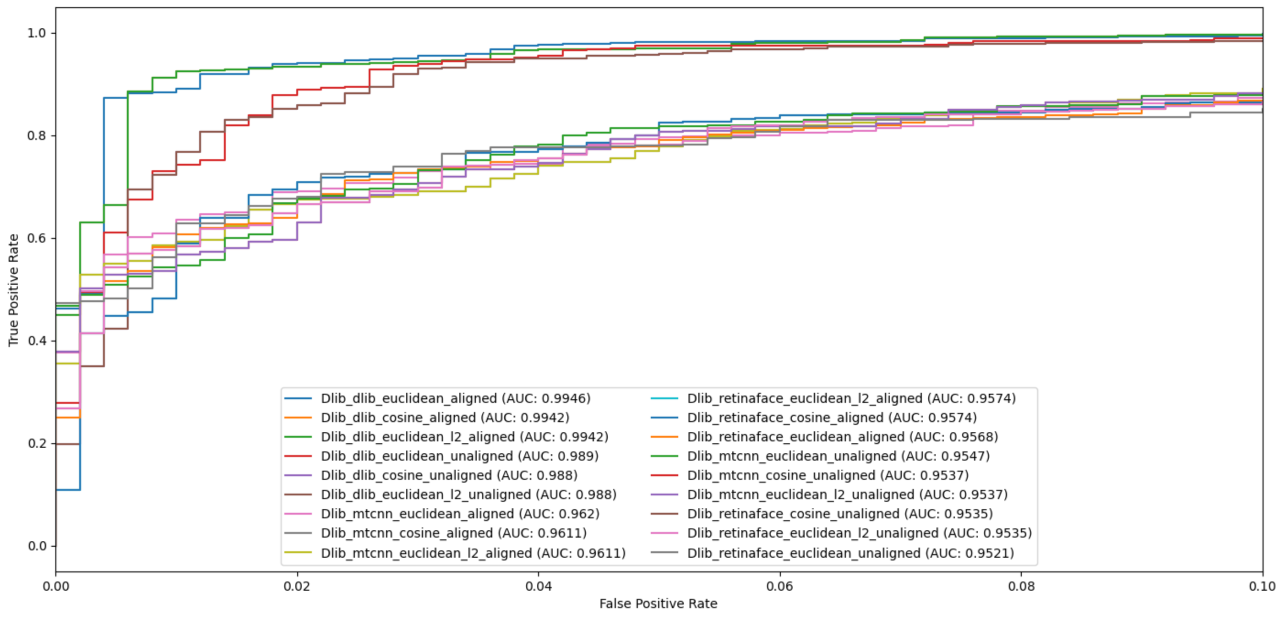

Figure 9. ROC Curves and AUC Scores for Dlib

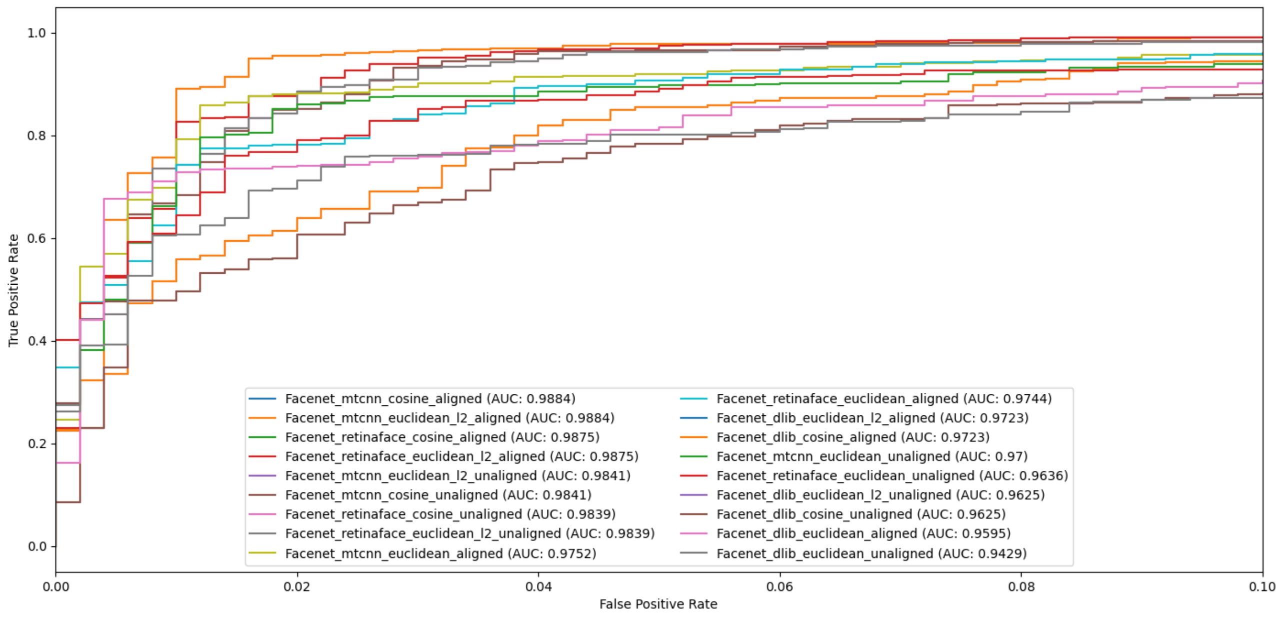

Figure 10. ROC Curves and AUC Scores for FaceNet128d

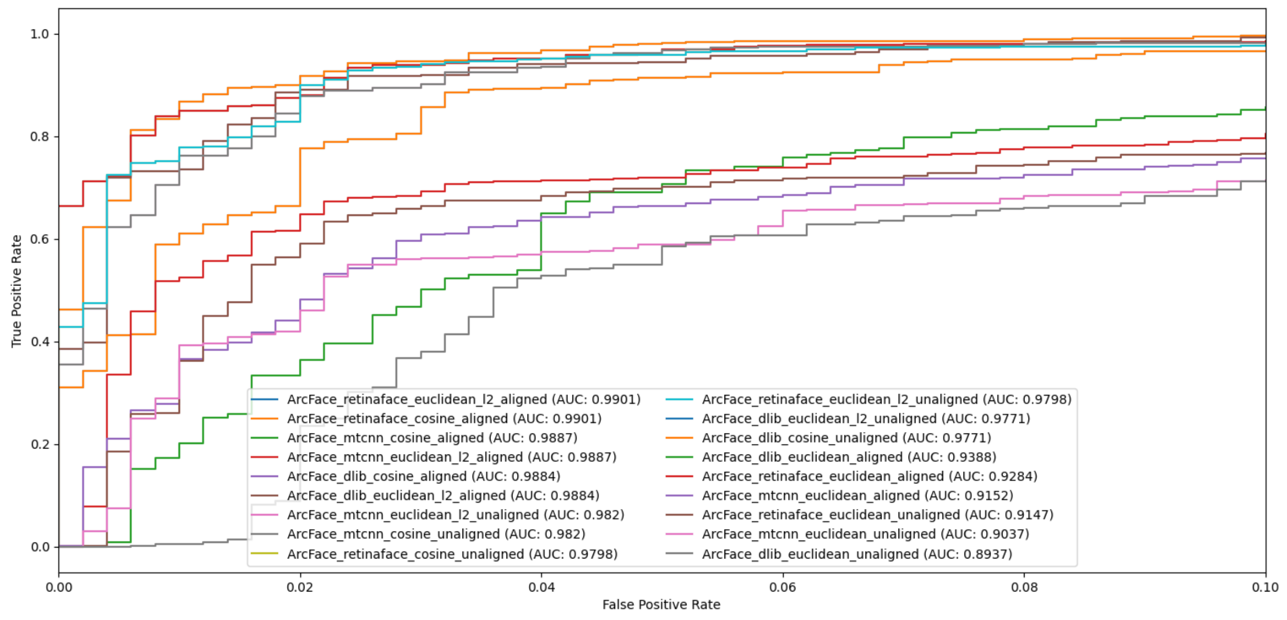

Figure 11. ROC Curves and AUC Scores for ArcFace

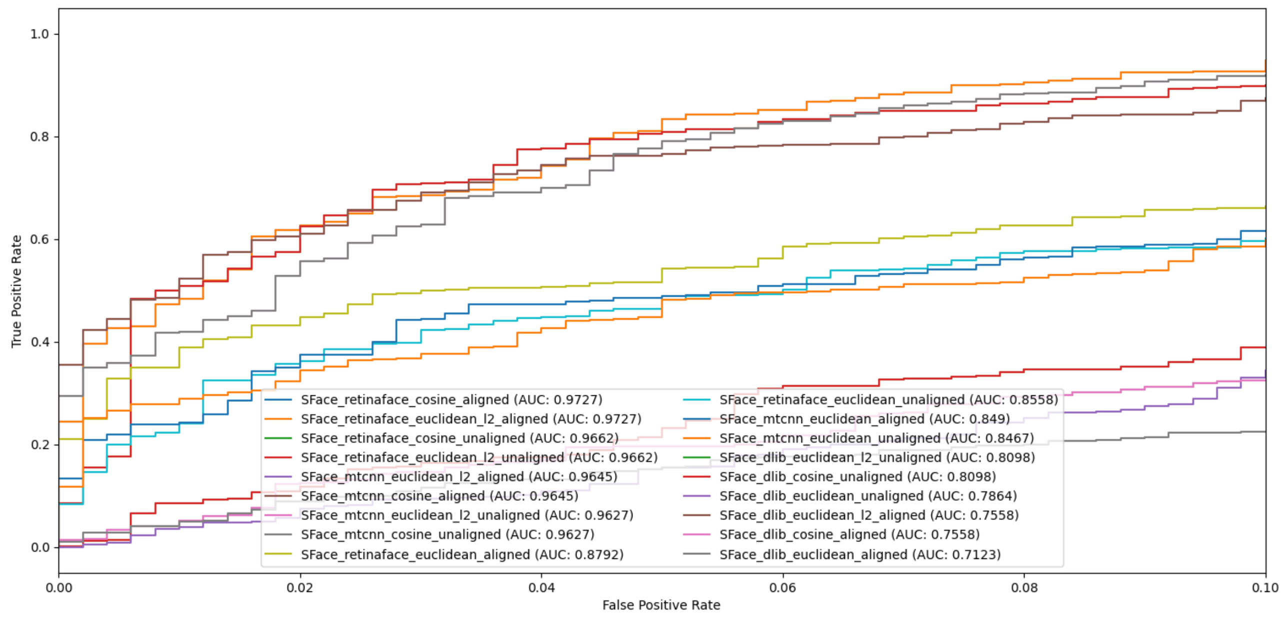

Figure 12. ROC Curves and AUC Scores for SFace

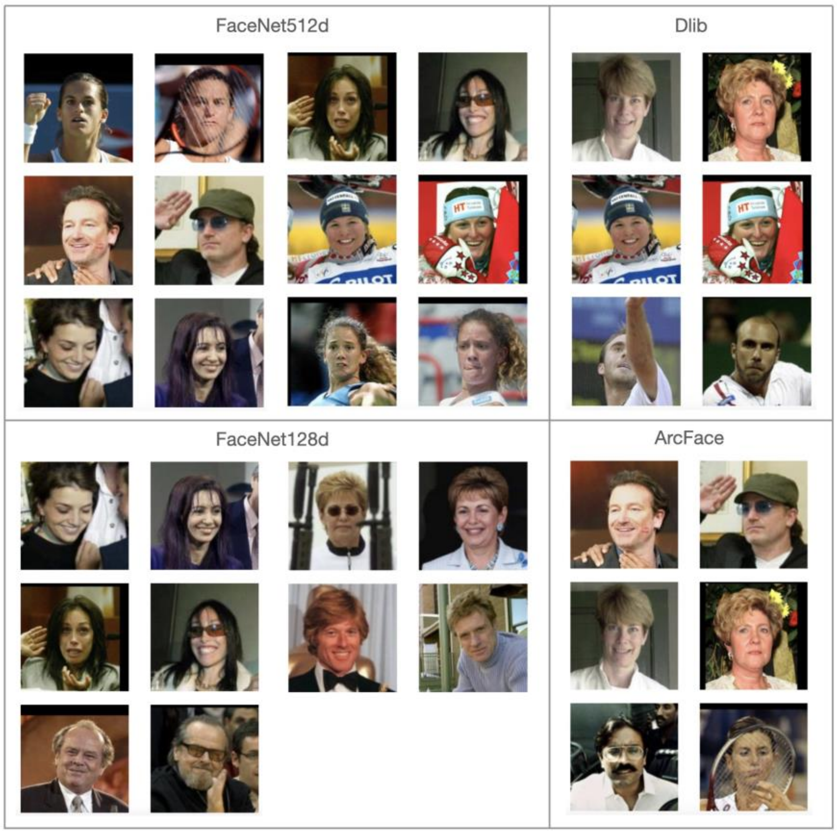

Figure 13. Common Misclassifications

REFERENCES

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deep-face: Closing the gap to human-level performance in face verification”, **In Proceedings of the IEEE conference on computer vision and pattern recognition**, 1701–1708, 2014.

[2] 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**, 815–823, 2015.

[3] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition”, **In British Machine Vision Conference**, 2015.

[4] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition”, **In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**, pages 4690–4699, 2019.

[5] D. E. King, “Dlib-ml: A machine learning toolkit”, *The Journal of Machine Learning Research*, 10, 1755–1758, 2009.

[6] Y. Zhong, W. Deng, J. Hu, D. Zhao, X. Li, and D. Wen, “Sface: Sigmoid-constrained hypersphere loss for robust face recognition”, *IEEE Transactions on Image Processing*, 30:2587–2598, 2021.

[7] B. Amos, B. Ludwiczuk, M. Satyanarayanan, et al. “Openface: A general-purpose face recognition library with mobile applications”, *CMU School of Computer Science*, 6(2):20, 2016.

[8] 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**, 1891–1898, 2014.

[9] S. I. Serengil and A. Ozpinar, “Lightface: A hybrid deep face recognition framework”, **In 2020 Innovations in Intelligent Systems and Applications Conference (ASYU)**, 23–27. IEEE, 2020.

[10] G. Bradski, “The opencv library”, *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg. “Ssd: Single shot multi-box detector”, **In European conference on computer vision**, 21–37. Springer, 2016.

[12] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, et al. “Mediapipe: A framework for building perception pipelines”, arXiv preprint arXiv:1906.08172, 2019.

[13] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann. “Blazeface: Sub-millisecond neural face detection on mobile gpus”, arXiv preprint arXiv:1907.05047, 2019.

[14] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. “Joint face detection and alignment using multitask cascaded convolutional networks”, *IEEE signal processing letters*, 23(10):1499–1503, 2016.

[15] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou. “Retinaface: Single-shot multi-level face localisation in the wild”. **In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**, pages 5203–5212, 2020.

[16] S. I. Serengil and A. Ozpinar. “Hyperextended lightface: A facial attribute analysis framework”, **In 2021 International Conference on Engineering and Emerging Technologies (ICEET)**, 1–4. IEEE, 2021.

[17] G. B Huang, M. Mattar, T. Berg, and E. L. Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments”, **In Workshop on faces in Real-Life Images: detection, alignment, and recognition**, 2008.

[18] O. Kramer, “Scikit-learn”, In Machine learning for evolution strategies, pages 45–53. Springer, 2016.

[19] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. “Attribute and simile classifiers for face verification”, **In 2009 IEEE 12th international conference on computer vision,** 365–372. IEEE, 2009.

[20] J. R. Quinlan, C4. 5: **programs for machine learning. Elsevier**, 2014.

[21] S. I. Serengil, Deepface: A lightweight face recognition and facial attribute analysis (age, gender, emotion and race) library for python, https://github.com/serengil/deepface, 15.04.2024.

[22] A. Z. Omkar, M. Parkhi, A. Vedaldi, Vgg face descriptor, https://www.robots.ox.ac.uk/∼vgg/software/vgg_face/, 15.04.2024.

[23] D. Sandberg, Facenet: Face recognition using tensorflow, https://github.com/davidsandberg/facenet, 15.04.2024.

[24] L. D Garse, Keras insightface, https://github.com/leondgarse/Keras_insightface, 15.04.2024.

[25] Y. Feng, SFace, https://github.com/opencv/opencv_zoo/tree/main/models/face_recognition_sface, 15.04.2024.

[26] V. S. Wang, Keras-openface2, https://github.com/iwantooxxoox/Keras-OpenFace, 15.04.2024

[27] S. Ghosh, Deepface, https://github.com/swghosh/DeepFace, 2019. 15.04.2024.

[28] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “Vggface2: A dataset for recognising faces across pose and age”, **In 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)**, 67–74. IEEE, 2018.

[29] R. Ran, Deepid implementation, https://github.com/Ruoyiran/DeepID, 15.04.2024.

[30] I. P. Centeno, Mtcnn, https://github.com/ipazc/mtcnn, 15.04.2024.

[31] S. Bertrand, Retinaface-tf2, https://github.com/StanislasBertrand/RetinaFace-tf2, 15.04.2024.

[32] S. I. Serengil, Retinaface: Deep face detection library for python, https://github.com/serengil/retinaface, 15.04.2024.

[33] K. Yildiz, E. Gunes, A. Bas, “CNN-based Gender Prediction in Uncontrolled Environments”, *Duzce University Journal of Science & Technology*, 890-898. 2021.

[34] H. Goze, O. Yildiz, “A New Deep Learning Model for Real-Time Face Recognition and Time Marking in Video Footage”, *Journal of Information Technologies*, 167-175. 2022.

[35] G. Guodong, N. Zhang, “A survey on deep learning based face recognition”, *Computer Vision and Image Understanding*, 189, 102805, 2019.

[36] M. Hassaballah, S. Aly, “Face recognition: challenges, achievements and future directions”, *IET Computer Vision*, 9(4), 614-626, 2015.

[37] D. Heinsohn, E. Villalobos, L. Prieto, D. Mery, “Face recognition in low-quality images using adaptive sparse representations”, *Image and Vision Computing*, 85, 46-58, 2019.

[38] P. J. Phillips, A. J. O'toole, “Comparison of human and computer performance across face recognition experiments”, *Image and Vision Computing*, 32(1), 74-85, 2014.

[39] E. G. Ortiz, B. C. Becker, “Face recognition for web-scale datasets”, *Computer Vision and Image Understanding*, 118, 153-170, 2014.