Face Recognition Based on MTCNN and FaceNet

Rongrong Jin, Hao Li, Jing Pan, Wenxi Ma, and Jingyu Lin

Abstract

Face recognition performance improves rapidly with the recent deep learning technique developing and underlying large training dataset accumulating. However, face images in the wild undergo large intra-personal variations, such as poses, illuminations, occlusions, and low resolutions, which cause great challenges to face-related applications.This paper addresses this challenge by proposing a deep learning framework which is based on MTCNN and FaceNet, which can recover the canonical view of face images. In our project, we build our own Face Recognition System, which achieves high accuracy on the LFW benchmark.We use the inherent correlation between detection and calibration to improve their performance under the multi-task framework of deep cascading. In particular, we use a three-tiered architecture combined with a well-designed roll neural network algorithm to detect faces and roughly locate key points.In the FaceNet method, it directly learns the mapping from a face image to a compact Euclidean space, where distance directly corresponds to a measure of facial similarity.Once this space is generated, face recognition, validation and clustering can be easily implemented using the standard FaceNet embedding technique as the feature vector.This approaches dramatically reduce the intra-person variances, while maintaining the inter-person discriminativeness. Maybe there is something not that perfect during our experiments, but we are going to summarize our experiments and present some challenges lying ahead in recent face recognition.

1. Introduction

With the rapid development of artificial intelligence in recent years, facial recognition gains more and more attention. Compared with the traditional card recognition, fingerprint recognition and iris recognition, face recognition has many advantages, including but limit to non-contact, high concurrency, and user friendly. It has high potential to be used in government, public facilities, security, e-commerce, retailing, education and many other fields.

Traditional face recognition methods use feature operators to model face, which is simple and easy to implement. However, with the further research, these algorithms can show strong effectiveness in finding linear structures, but when facing potential nonlinear structures, they often achieve unsatisfactory recognition results.

With the development of deep learning and the introduction of deep convolutional neural networks, the accuracy and speed of face recognition have made great strides. However, the results from different networks and models are very different. Previous face recognition approaches based on deep networks use a classification layer(Taigman et al. 2014; Tang 2015), they regard face recognition as a classification task. The number of softmax output is the number of face tags. Therefore, every time a new sample comes in, the whole model needs to be retrained. While FaceNet directly trains its output to be a compact 128-D embedding using a triplet-based loss function based on LMNN(Schroff, Kalenichenko, and Philbin 2015). The triplets consist of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the positive pair. The thumbnails are tight crops of the face area, no 2D or 3D alignment, other than scale and translation is performed. The benefit of this approach is much greater representational efficiency: they achieve state-of-the-art face recognition performance using only 128-bytes per face. So we use FaceNet, 128 dimensional vector to represent face, and then recognize face by calculating vector distance.

In order to achieve better performance, we first use MTCNN(Zhang et al. 2016) to do face detection. Then use the result of MTCNN as the input of FaceNet to perform face recognition. MTCNN network, which is a mainstream target detection network with high detection accuracy, lightweight and real-time.

So our face recognition process is mainly divided into two steps: face detection and face recognition. Firstly, MTCNN is used for face detection to get accurate face coordinates. Based on the results of the previous step, FaceNet is used for face recognition. The processing flow of MTCNN is as follows: First of all, the test image is continuously resized to get the image pyramid. Then the image pyramid is input into P-Net to get a large number of candidates. The candidate images screened by P-Net are fine tuned by R-Net. After many candidates are removed by R-Net, the images are input to O-Net. Finally, the accurate bbox coordinates are output. Compared with DeepFace, FaceNet retains face alignment, abandons feature extraction steps, and directly uses CNN to train end-to-end after face alignment.

2. Related Work

Face detection

Face detection are essential to many face applications, such as face recognition and facial expression analysis. However, the large visual variations of faces, such as occlusions, large pose variations and extreme lightings, impose great challenges for these tasks in real world applications.

The cascade face detector proposed by Viola and Jones(Viola and Jones 2004) utilizes Haar-Like features and AdaBoost to train cascaded classifiers, which achieves good performance with real-time efficiency. However, quite a few works(Yang et al. 2014; Pham et al. 2010) indicate that this kind of detector may degrade significantly in real-world applications with larger visual variations of human faces even with more advanced features and classifiers. Besides the cascade structure(Zhu and Ramanan 2012), introduce deformable part models (DPM) for face detection and achieve remarkable performance. However, they are computationally expensive and may usually require expensive annotation in the training stage. Recently, convolutional neural networks (CNNs) achieve remarkable progresses in a variety of computer vision tasks, such as image classification and face recognition(Sun, Wang, and Tang 2014). Inspired by the significant successes of deep learning methods in computer vision tasks, several studies utilize deep CNNs for face detection. Yang et al.(Yang et al. 2016) train deep convolution neural networks for facial attribute recognition to obtain high response in face regions which further yield candidate windows of faces. However, due to its complex CNN structure, this approach is time costly in practice. Li et al.(Li et al. 2015) use cascaded CNNs for face detection, but it requires bounding box calibration from face detection with extra computational expense and ignores the inherent correlation between facial landmarks localization and bounding box regression.

Face recognition

Using deep neural networks to learn effective feature representations has become popular in face recognition(Sun, Wang, and Tang 2013). With better deep network architectures and supervisory methods, face recognition accuracy has been boosted rapidly in recent years. Previous face recognition approaches based on deep networks use a classification layer (Taigman et al. 2014) trained over a set of known face identities and then take an intermediate bottleneck layer as a representation used to generalize recognition beyond the set of identities used in training. The downsides of this approach are its indirectness and its inefficiency: one has to hope that the bottleneck representation generalizes well to new faces; and by using a bottleneck layer the representation size per face is usually very large (1000s of dimensions). Some recent work has reduced this dimensionality using PCA, but this is a linear transformation that can be easily learnt in one layer of the network.

3. Method

For accurate face recognition, we train two networks, MTCNN and FaceNet. MTCNN is used to detect the face and get the exact coordinates of the face. Based on the results of face detection, face recognition is performed using FaceNet.

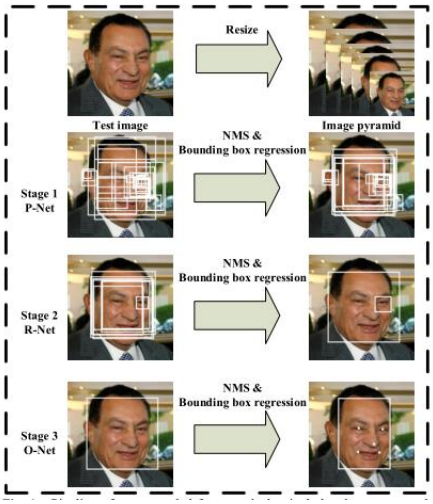

Figure 1: Pipeline of MTCNN cascaded framework that includes three-stage multi-task deep convolutional networks. Firstly, candidate windows are produced through a fast Proposal Network (P-Net). After that, we refine these candidates in the next stage through a Refinement Network (R-Net). In the third stage, the Output Network (O-Net) produces final bounding box.

.

FaceNet directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.

3.1. MTCNN

MTCNN is a deep cascaded multi-task framework which exploits the inherent correlation between detection and alignment to boost up their performance. The framework of MTCNN leverages a cascaded architecture with three stages of carefully designed deep convolutional networks to predict face and landmark location in a coarse-to-fine manner. In addition, a new online hard sample mining strategy that further improves the performance in practice.

3.1.1. Overall Framework

The overall pipeline of MTCNN is shown in Figure. 1. Given an image, we initially resize it to different scales to build an image pyramid, which is the input of the following three-stage cascaded framework:

Stage 1: We exploit a fully convolutional network, called Proposal Network (P-Net), to obtain the candidate facial windows and their bounding box regression vectors. Then candidates are calibrated based on the estimated bounding box regression vectors. After that, we employ non-

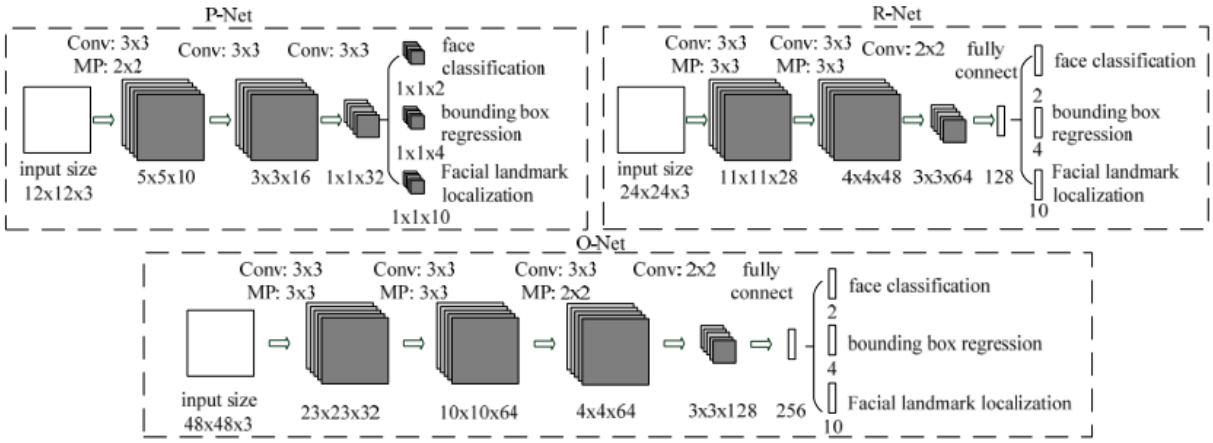

Figure 2: The architecture of P-Net, R-Net, and O-Net. Where "MP" means max pooling and "Conv" means convolution. The step size in convolution and pooling is 1 and 2, respectively

maximum suppression (NMS) to merge highly overlapped candidates.

Stage 2: All candidates are fed to another CNN, called Refine Network (R-Net), which further rejects a large number of false candidates, performs calibration with bounding box regression, and conducts NMS.

Stage 3: This stage is similar to the second stage, but in this stage we aim to identify face regions with more supervision. In particular, the network will output five facial landmarks' positions.

3.1.2. CNN Architectures

We use 3×3 filter rather than 5×5 filter to reduce the computing while increase the depth to get better performance. With these improvements, compared to the previous architecture in(Li et al. 2015), we can get better performance with less runtime. The CNN architectures are shown in Figure. 2. We apply PReLU(He et al. 2015) as nonlinearity activation function after the convolution and fully connection layers(except output layers).

3.1.3. Training

We leverage three tasks to train our CNN detectors: face/non-face classification, bounding box regression, and facial landmark localization.

1) Face classification: The learning objective is formulated as a two-class classification problem. For each sample x_i , we use the cross-entropy loss:

$$L_i^{det} = -(y_i^{det} log(p_i) + (1 - y_i^{det})(1 - log(p^i))) \quad (1)$$

where p_i is the probability produced by the network that indicate sample x_i being a face. The notation $y_i^{det} \in \{0, 1\}$ denotes the ground-truth label.

2) Bounding box regression: For each candidate window, we predict the offset between it and the nearest ground truth. The learning objective is formulated as a regression problem, and we employ the Euclidean loss for each sample x_i:

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2 \quad (2)$$

where $\hat{y}_i^{box}$ is the regression target obtained from the network and y_i^{box} is the ground-truth coordinate.

3) Facial landmark localization: Similar to bounding box regression task, facial landmark detection is formulated as a regression problem and we minimize the Euclidean loss:

$$L_i^{landdmark} = \left\| \hat{y}_i^{landmark} - y_i^{landmark} \right\|_2^2 \quad (3)$$

where $\hat{y}_i^{landmark}$ is the facial landmark's coordinates obtained from the network and $y_i^{landmark}$ is the ground-truth coordinate for the i-th sample.

4) Multi-source training: Since we employ different tasks in each CNN, there are different types of training images in the learning process, such as face, non-face, and partially aligned face. In this case, some of the loss functions (i.e., Eq. (1)-(3)) are not used. The overall learning target can be formulated as:

$$min \sum_{i-1}^{N} \sum_{j \in U} \alpha_j \beta_i^j L_i^j \quad (4)$$

where $U = \{det, box, landmark\}$, and N is the number of training samples and aj denotes on the task importance.

Figure 3: FaceNet model structure.

.

3.2. FaceNet

FaceNet is adopted in our face recognition truncation. FaceNet directly trains its output to be a compact 128-D embedding using a tripletbased loss function based on LMNN. Our triplets consist of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the positive pair from the negative by a distance margin. The thumbnails are tight crops of the face area, no 2D or 3D alignment, other than scale and translation is performed.And it is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity: faces of the same person have small distances and faces of distinct people have large distances.

3.2.1 End-to-end learning

Instead of using the traditional softmax method to do classification learning, FaceNet extracted a certain layer as a feature to learn a coding method from the image to The European space, and then do face recognition face verification and face clustering based on this code.Given the model details, and treating it as a black box (see Figure 3), the most important part of our approach lies in the end-to-end learning of the whole system. To this end we employ the triplet loss that directly reflects what we want to achieve in face verification, recognition and clustering. Namely, we strive for an embedding f(x), from an image x into a feature space Rd, such that the squared distance between all faces, independent of imaging conditions, of the same identity is small, whereas the squared distance between a pair of face images from different identities is large.

3.2.2 Triplet Loss

The triplet loss is more suitable for face verification. The motivation is that the loss from(Sun, Wang, and Tang 2014) encourages all faces of one identity to be projected onto a single point in the embedding space. The triplet loss, however, tries to enforce a margin between each pair of faces from one person to all other faces. This allows the faces for one identity to live on a manifold, while still enforcing the distance and thus discriminability to other identities.

The embedding is represented by $f(x) \in R^d$. It embeds an image x into a d-dimensional Euclidean space. Additionally, we constrain this embedding to live on the d-dimensional hypersphere, i.e.$\|f(x)_2\| = 1$ This loss is motivated in(Weinberger 2009) in the context of nearest-neighbor classification. Here we want to ensure that an image x_i^a (anchor) of a specific person is closer to all other images x_i^p (positive) of the same person than it is to any image x_i^n (negative) of any other person. This is visualized in Figure 4.

Thus we want,

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \qquad (5)$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in T \qquad (6)$$

where α is a margin that is enforced between positive and negative pairs. T is the set of all possible triplets in the training set and has cardinality N. The loss that is being minimized is then L =

$$\sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+ \qquad (7)$$

Generating all possible triplets would result in many triplets that are easily satisfied. These triplets would not contribute to the training and result in slower convergence, as they would still be passed through the network.

Figure 4: The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity.

3.2.3 Triplet Selection

In order to ensure fast convergence it is crucial to select triplets that violate the triplet constraint in Eq. (5). This means that, given x_i^a , we want to select an x_i^p (hard positive) such that $argmax_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$ and similarly x_i^n (hard negative) such that $argmax_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2$.

It is infeasible to compute the argmin and argmax across the whole training set. Additionally, it might lead to poor training, as mislabelled and poorly imaged faces would dominate the hard positives and negatives. There are two obvious choices that avoid this issue:

• Generate triplets offline every n steps, using the most recent network checkpoint and computing the argmin and argmax on a subset of the data.

• Generate triplets online. This can be done by selecting the hard positive/negative exemplars from within a mini-batch.

Instead of picking the hardest positive, we use all anchorpositive pairs in a mini-batch while still selecting the hard negatives. We don't have a side-by-side comparison of hard anchor-positive pairs versus all anchor-positive pairs within a mini-batch, but we found in practice that the all anchorpositive method was more stable and converged slightly faster at the beginning of training. Selecting the hardest negatives can in practice lead to bad local minima early on in training, specifically it can result in a collapsed model (i.e. f(x) = 0). In order to mitigate this, it helps to select x_i^n such that

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2 \qquad (8)$$

To sum up, Correct triplet selection is crucial for fast convergence. On the one hand we would like to use small

mini-batches as these tend to improve convergence during Stochastic Gradient Descent (SGD). On the other hand, implementation details make batches of tens to hundreds of exemplars more efficient.

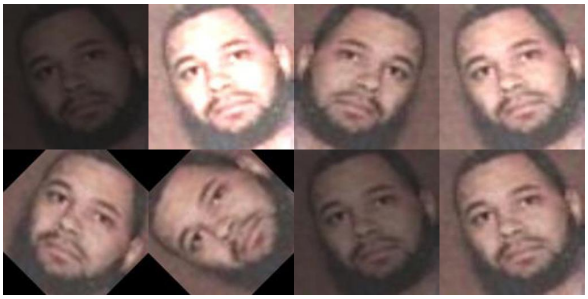

Figure 5: Enhanced image and the original image. The first line is enhanced by four contrast changes. The second line is the enhanced image with random operations and the original image

.

3.3. Data Augmentation

Large-scale datasets are the prerequisite for the successful application of deep neural networks. The image augmentation technology uses a series of random changes to the training images to generate similar but different training samples, thereby expanding the size of the training dataset.

Another explanation for image augmentation is that randomly changing the training samples can reduce the model's dependence on certain attributes and improve the generalization ability of the model. For example, we can crop the image in different ways to make the objects of interest appear in different positions, thereby reducing the dependence of the model on the position of the object. We can also adjust factors such as contrast ratio to reduce the model's sensitivity to brightness.

In order to enhance the robustness of the model when predicting, we decide to apply image augmentation to datasets with random operations when training. The methods we use here are: random fixed ratio cropping, mirror flipping, turning left 45°, turning right 45°, etc. Some samples see in Figure 5.

4. Experiments

4.1. MTCNN Backbone networks

Before the experiment, we notice the performance of multiple CNNs might be limited by the following facts:

(1) Some filters in convolution layers lack diversity that may limit their discriminative ability.

(2) Considering face detection is a challenging binary classification task, so it may need less numbers of filters per layer. To this end, we reduce the number of filters and change the 5×5 filter to 3×3 filter to reduce the computing while increase the depth to get better performance. With these improvements, compared to the previous architecture, we can get better performance with less runtime which are shown in Table I.

Table 1: COMPARISON OF SPEED AND VALIDATION ACCURACY OF OUR CNNs AND PREVIOUS CNNs

Group	CNN	300xForward Propagation	Validation Accuracy
Group1	12-Net	0.043s	93.10%
	P-Net	0.040s	93.70%
Group2	24-Net	0.738s	93.80%
	R-Net	0.466s	94.50%
Group3	48-Net	3.601s	92.10%
	O-Net	1.411s	93.50%

So in the MTCNN part, with the cascade structure, our method can achieve high speed in joint face detection and alignment. We compare our method with some classic techniques on GPU and the results are shown in Table II.

Table 2: SPEED COMPARISON OF OUR METHOD AND OTHER METHODS

Method	GPU	Speed
Ours	NIVIDIA Titan Black	93FPS
Casacade CNN	NIVIDIA Titan Black	100FPS
Faceness	NIVIDIA Titan Black	20FPS
DP2MFD	NIVIDIA Tesla K20	0.285FPS

4.2. FaceNet Deep Architecture

We use three backbone networks, which are ZeilerFergus with 1×1 convolution and norm, ResNet50, and ResNet101. Below we use NN1, NN2, and NN3 to replace them respectively as is shown in Table 3. Among them, NN1 uses model pre-trained on ImageNet, and the other two backbone networks do not use pre-trained models. In NN1, we adds 1×1×d convolutional layers, between the standard convolutional layers of the ZeilerFergus architecture and results in a model 22 layers deep. It has a total of 140 million parameters and requires around 1.6 billion FLOPS per image. We retain the original architecture for the other two backbone networks.

4.3. Performance on LFW

We evaluate our model on LFW using the standard protocol for unrestricted, labeled outside data. Nine training splits are used to select the L2-distance threshold. Classification (same or different) is then performed on the tenth test split. The selected optimal threshold is 1.242 for all test splits except split eighth (1.256). We achieve a classification accuracy of **89.52%±0.18**, **90.16%±0.15** and **92.86%±0.12** when using NN1, NN2 and NN3 respectively.

layer	size-in	size-out	1ayer kernal	param	FLPS
conv1	220x220x3	110x110x64	7x7x3,2	9K	115M
pool1	110x110x64	55x55x64	3x3x64,2	0	
rnorm1	55x55x64	55x55x64		0	
conv2a	55x55x64	55x55x64	1x1x64,1	4K	13M
conv2	55x55x64	55x55x192	3x3x64,1	111K	135M
rnorm2	55x55x192	55x55x192		0	
pool2	55x55x192	28x28x192	3x3x192,2	0	
conv3a	28x28x192	28x28x192	1x1x192,1	37K	29M
conv3	28x28x192	28x28x384	3x3x192,1	664K	521M
pool3	28x28x384	14x14x384	3x3x384,2	0	
conv4a	14x14x384	14x14x384	1x1x384,1	148K	29M
conv4	14x14x384	14x14x256	3x3x384,1	885K	173M
conv5a	14x14x256	14x14x256	1x1x256,1	66K	13M
conv5	14x14x256	14x14x256	3x3x256,1	590K	116M
conv6a	14x14x256	14x14x256	3x3x256,1	66K	13M
conv6	14x14x256	14x14x256	3x3x256,1	590K	116M
pool4	14x14x256	7x7x256	3x3x256,2	0	
concat	7x7x256	7x7x256		0	
fc1	7x7x256	1x32x128	maxout p=2	103M	103M
fc2	1x32x128	1x32x128	maxout p=2	34M	34M
fc7128	1x32x128	1x1x128		524K	0.5M
L2	1x1x128	1x1x128		0	
total				140M	1.6B

Table 3: FaceNet Deep Architectures. This table compares the performance of the different backbones we used on the LFW dataset. NN1 is ZeilerFergus with 1×1 convolution and norm, NN2 is ResNet50 , and NN3 is ResNet101. Reported are the mean validation rate VALs at 10E-3 false accept rate. The input image size is set to 160×160.

.

5. Conclusion

In this paper, we have proposed a multi-task cascaded CNNs based framework combine with a unified embedding for face detection and recognition. Ex-perimental results demonstrated that the method linger around the state-of-the-art methods across challenging AFLW benchmark for face alignment. The three main contributions for performance improvement are carefully de-signed cascaded CNNs architecture, online hard sample mining strategy, and joint face alignment learning.

What's more, We use the a unified embedding method to directly learn an embedding into an Euclidean space for face verification.This sets it apart from other methods who use the CNN bottleneck layer, or require additional post-processing such as concate-nation of multiple models and PCA, as well as SVM classification. Our end-to-end training both simplifies the setup and shows that directly optimizing a loss relevant to the task at hand improves performance.

Three different recognition tricks chosen for their out-performed results to our work were as follows: end-to-end learning, triplet loss, and data augmentation; All three combined with one another by convolutional neural networks that return the main features of detected faces.

Future work will focus on better understanding of the error cases, further improving the model, and also reducing model size and reducing CPU requirements. We will also look into ways of improving the currently extremely long training times, e.g. variations of our curriculum learning with smaller batch sizes and offline as well as online positive and negative mining.

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Li, H.; Lin, Z.; Shen, X.; Brandt, J.; and Hua, G. 2015. A convolutional neural network cascade for face detection. In *Computer Vision Pattern Recognition*.

Pham, M. T.; Gao, Y.; Hoang, V. D. D.; and Cham, T. J. 2010. Fast polygonal integration and its application in extending haar-like features to improve object detection. In *Computer Vision and Pattern Recognition*.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sun, Y.; Wang, X.; and Tang, X. 2013. Hybrid Deep Learning for Face Verification. In *IEEE International Conference on Computer Vision*.

Sun, Y.; Wang, X.; and Tang, X. 2014. Deep Learning Face Representation by Joint Identification-Verification. *Advances in neural information processing systems* 27.

Taigman, Y.; Yang, M.; Ranzato, M.; and Wolf, L. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tang, Y. S. W. 2015. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Viola, P.; and Jones, M. J. 2004. Robust Real-Time Face Detection. *International Journal of Computer Vision* 57(2): 137–154.

Weinberger, K. Q. 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Jmlr* 10.

Yang, B.; Yan, J.; Lei, Z.; and Li, S. Z. 2014. Aggregate channel features for multi-view face detection .

Yang, S.; Luo, P.; Loy, C. C.; and Tang, X. 2016. From Facial Parts Responses to Face Detection: A Deep Learning Approach. In *IEEE International Conference on Computer Vision*.

Zhang, K.; Zhang, Z.; Li, Z.; and Qiao, Y. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23(10): 1499–1503. doi:10.1109/LSP.2016.2603342.

Zhu, X.; and Ramanan, D. 2012. Face Detection, Pose Estimation, and Landmark Localization in the Wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*.