Real-Time Deep-Learning-Based System for Facial Recognition

Wesley L. Passos[1], Igor M. Quintanilha[1], Gabriel M. Araujo[2].

***Abstract*— This work presents an open code deep-learning-based system to perform facial recognition. The system is composed of five main steps: face segmentation, facial features detection, face alignment, embedding, and classification. We use using deep learning methods for the fiducial points extraction and embedding. Support Vector Machine (SVM) is used for classification task since it is fast for both training and inference. The system achieves an error rate of 0.12103 for facial features detection, which is pretty close to state of the art algorithms, and 0.05 for face recognition. Besides, it is capable to run in real-time.**

***Keywords*— Face recognition, Deep learning, Facial features, Neural networks, Pattern classification.**

I. Introduction

The computational power evolution, big data availability, and the social media spread led to a considerable expansion in facial recognition area. In the past a forensic tool, nowadays facial recognition algorithms are now part of daily people's lives. For instance, social networks, mobile devices unlock systems, among others.

This work proposes a deep learning [1] based face recognition system. It uses a type of deep neural network specially developed to computer vision applications: *Convolutional Neural Networks* (CNN). CNNs, also known as ConvNets, are networks which use convolutional layers as feature extractors [2].

The proposed system is mainly composed of five parts: face segmentation, facial features detection (using deep learning), face alignment, codification, and classification (using Support Vector Machine – SVM). Each step is detailed in Section III. However, facial key points (also known as facial landmarks or facial fiducial points) detection is shown in greater details, since it is essential in many other applications. We also highlight the classification step, which performs the subject identification.

II. Related work

Nowadays, the most common approaches to performing facial features detection and tracking as well as subject recognition are machine learning based. These methods can be categorized as *shallow* or *deep*.

Shallow methods use descriptors such as *Scale-Invariant Feature Transform* (SIFT), *Local Binary Patterns* (LBP) and *Histogram of Oriented Gradient* (HOG) [3], [4], [5] which are aggregated to compose a global face descriptor using a pooling mechanism, such as the Fisher vector [6].

[1]PEE/COPPE Universidade Federal do Rio de Janeiro, Cx. P. 68504, Rio de Janeiro, RJ, 21945-970, Brasil. [2]Centro Federal de Educação Tecnológica, Nova Iguaçu, RJ, 26041-271, Brasil. E-mails: wesley.passos, igor.quintanilha@smt.ufrj.br, gabriel.araujo@cefet-rj.br

Modern deep methods use Convolutional Neural Networks (CNN) as nonlinear feature extractors. A famous example is the *DeepFace* [7], that uses a CNN architecture known as a Siamese network. This architecture compares Euclidean distances between descriptors obtained by the output of this CNN when pairs of faces images pass through it. The training objective is to minimize the distance between pairs of congruent faces (same identity) and maximize the distances between the incongruous ones (different identities). Also, *DeepFace* uses CNNs committee in a preprocessing step in which the faces are aligned to a canonical pose using a 3D model. In its most recent version [8], *DeepFace* database include 10 million identities and 50 images per identity.

Another deep-network-based method that worth mention is the *FaceNet* [9]. It uses a huge dataset with 200 million identities and 800 million image pairs to train a CNN similar to [10], [11]. This method has better performance in *Labeled Faces in the Wild* (LFW) and *YouTube Faces* (YTF) datasets.

Currently *DeepFace* [7] and *FaceNet* [9] are the CNN-based systems with higher accuracy in the literature. However, they are trained using private dataset containing millions of images from social networks. These datasets are considerably greater than the ones available to research [12]. So, to build an open source code system which can be used in mobile devices, the authors of [12] published the *OpenFace*. It is a general purpose library to perform face recognition based on *DeepFace* and *FaceNet*.

This work presents an individual classification system based on deep learning similar to the one in [12]. However, the proposed system was designed to be used in real-time applications. So, the proposed architecture uses fewer parameters as possible to not compromise the frame rate. The classification step is based on *Support Vector Machine* (SVM) with a linear kernel. It is different from the one in [12] since we use deep learning to detect facial features. It differs from [7], [9] because it is open source, under MIT license, and we use public datasets to train and validate our system.

III. Proposed system

In this section, we describe the proposed system. For better understanding and facilitate the implementation, the process is separated into five blocks as illustrated in Fig. 1. Each part uses computer vision and machine learning techniques to perform its task.

Fig. 1. Simplified block diagram of the proposed system.

A. Face detection

Before recognizing a person in an image, it is necessary to locate its face. Although a classical technique from 2004 known as Viola-Jonas [13] has been broadly used, better techniques using deep learning are rising [14]. This work uses a Histogram of Oriented Gradient (HOG) based method, the *Max-Margin Object Detection* (MMOD), implemented by using `dlib` library [15]. The HOG [16] is a scale, rotation and illumination invariant descriptor and has been mostly applied in image processing and computer vision applications. MMOD, which is similar to an SVM, is trained in HOG feature space to detect objects with high accuracy [15]. The segmented face is delivered to the facial feature extraction step.

B. Facial features extraction

In this step, a regressor is employed to extract critical facial key points on eyes, eyebrow, nose, lips, and others. There are many ways to perform this task, but the computer vision based methods are less expensive and intrusive. Nowadays, the algorithms using deep learning have the better results. This work tests the use of *Multi-Layer Perceptron* (MLP) and Convolutional (CNN) neural networks, described as follows.

1) Multi-Layer Perceptron Neural Network: In a MLP network, with two layers, with N_1 neurons in the hidden layer and N neurons at the output, for an input $\boldsymbol{x} \in \mathbb{R}^{(C*H*W)}$ representing an image with C channels, height H and width W, the output is:

$$\boldsymbol{h} = \boldsymbol{W}_{xh}\boldsymbol{x} + \boldsymbol{w}_h, \quad z = \sigma(h), \quad \boldsymbol{y} = \boldsymbol{W}_{zl}\boldsymbol{z} + \boldsymbol{w}_l, \quad (1)$$

where $\boldsymbol{W}_{xh} \in \mathbb{R}^{(C*H*W)\times N_1}$ and $\boldsymbol{w}_h \in \mathbb{R}^{N_1}$ are the weights at the input layer, $\sigma(\cdot)$ is a differentiable nonlinear function, $\boldsymbol{W}_{zl} \in \mathbb{R}^{N_1 \times N}$ and $\boldsymbol{w}_l \in \mathbb{R}^N$ are the weights at the output layer. The training consists of minimizing a cost function with respect to the network parameters. Since it is a nonlinear problem, it is highly recommended optimizing the function by using small *batches* from the training set [1].

2) Convolutional Neural Network: MLP networks do not take spacial information from images into account to reduce the number of parameters. Since CNNs use filters to find local structures (e.g., edges in images), they have been applied in many image processing applications and other areas in which data has high dimension and many spacial structures. CNNs perform this task with less computational cost and parameters than MLPs. CNNs usually have two main components: *convolutional layers* and *pooling layers*.

CNNs operate over tensors rather than vectors and do so through convolution, that is the reason why they are called convolutional networks. A convolutional layer is composed of F filters $\boldsymbol{f}_f$ of the same size, with width W_f, height H_f and the same number of input channels. A tensor with all parameters has a model $\boldsymbol{F} \in \mathbb{R}^{F\times C\times H_f \times W_f}$. It is common to use $W_f = H_F = K$ (kernel size) with values $K = 3, 5$, or 7. The inner product between each filter $\boldsymbol{f}_f$ and each position of the image generates a point in the *features map* $\boldsymbol{a}_f$. These maps are stacked to compose the output $\boldsymbol{a} \in \mathbb{R}^{F\times H_0 \times W_0}$ whose width and height depend on the size of the input (image), the *padding* P and the *stride* S. This process is illustrated in Fig. 2.

The pooling layer has an important role in CNN since it aggregates information. Similarly to the convolutional layers, it has a kernel size, padding, and stride. However, it does not have weights as it only applies the same function and operates on each channel independently. The most common types of pooling are *max-pooling* (maximum value inside a window) and *average pooling* (average value inside a window).

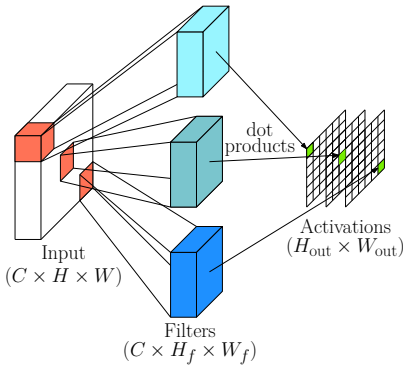

Fig. 2. Convolutional layer.

Usually, the CNNs architectures are composed of blocks with convolutional layers, a nonlinear activation function (e.g., ReLU [17]), regularization layer (e.g., dropout [18] or *batch-norm* [19]) and periodic pooling layer between the blocks. At the end, an MLP network, also known as Fully Connected (FC) layer, or a *global average pooling* layer [20] is added to generate the output. Many hyperparameters must be chosen, such as the number of layers, kernel size, stride, etc.

As for many machine learning algorithms, there is no a recipe to build a ConvNet. However, some architectures became popular: LeNet, AlexNet, VGG, ResNets and the DenseNets [1]. It is worthy to mention that ResNets is employed in many state-of-the-art computer vision algorithms due to its simplicity and high generalization capability. In this work, we use this CNN architecture due to the properties mentioned.

C. Face alignment

The objective of this step is to standardize the input of the classifier to generate a simpler classification model. Since the facial features location is known, it consists of aligning the faces from the image in such way that the nose, eyes, and mouth are aligned with the center of the image as better as possible. To do so, an affine transformation is employed. The affine transform does not distort the relative positions of the facial landmarks, i.e., parallel straight lines remain parallel after transformation. The `getAffineTransform` function, from OpenCV library, returns the rotation and translation necessary to take the original points to the desired ones (an average mask calculated from the points in training set). The `warpAffine` function, also from OpenCV library, applies the transformation, which also scales the resulting image.

D. Embedding

Given the aligned images, the next step consists of recognizing the person that is in the current image. However, feeding the classifier with the raw pixels of the image is not efficient. Therefore, an embedding process is employed. It consists of

extracting information from each image and creates an array, in a lower dimensional space, which better describes this image.

At this point, another deep learning method is employed to extract such array. Differently from the previous networks, the training objective is to minimize the so-called *triplet loss* [9] – at each iteration, the network is fed with three images: two distinct images of the same person and an image of a different person. The triplet loss is defined as

$$\sum_{i}^{N}[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+, \qquad (2)$$

where $f(x) \in \mathbb{R}^d$ is the function that transforms the image into an array of representations, x_i^a is an image of a person, x_i^p is another image from the same person, x_i^n is an image from a different person, α is a margin to be forced between the positive and negative pairs.

The network is then trained to create representation arrays that are close to images of the same people and as far as possible for different people. By repeating this process thousands of times for hundreds of thousands of images containing tens of thousands of people, the network is capable of learning how to generate good embedded representation for each person.

E. Classification

Due to the alignment and coding processes, the last step was greatly simplified. Ideally, images of the same person have similar feature vectors while images from different people have different ones. Therefore, in this last step, a simple machine learning algorithm is applied to classify each vector as belonging to a person or not. The SVM algorithm was chosen because it is fast for both training and inference.

IV. Experimental Procedures and Results

In this section, we report the experimental procedures and results obtained for fiducial points extraction and people classification.

A. Facial features extraction

All architectures used in these experiments are described in Section III-B. To have a fair comparison, all hyperparameters have been individually adjusted so that only the best results from each architecture are reported. The objective comparison metric used in all cases is the NRMSE (Section IV-A.2), and the performance of each algorithm is also evaluated subjectively through the analysis of the images and the fiducial points found.

1) Dataset: The dataset used for the fiducial points detection was introduced by the 300 faces in-the-wild [21] (300-W) challenge, which contains photos of people in different poses, illuminations, and expressions. For each image in the dataset there are 68 annotated points as shown in Fig. 3. Formerly known datasets, such as LFPW [22], AFW [14], and Helen [23] were re-annotated to follow this pattern and compose the dataset for the 300-W challenge. The training set consists of 811 training images from LFPW dataset, 337 images from AFW dataset, and 2000 training images from Helen dataset, totaling 3148 labeled images.

Fig. 3. Image examples from dataset used fiducial points detection.

The test set is split into three parts: the *common* test set, which composes the 300-W challenge public test set; the *challenging* test set; and the *300-W* challenge private test set. The common test set has 224 images of the LFPW test set and 330 images of the Helen test set, totaling 554 labeled images. The challenging test set is composed of the iBUG [21] dataset, which contains a more significant quantity of expressions and poses compared to the common test set, totaling 135 labeled images. Finally, the last test set (called here 300-W) is composed by the 300-W dataset, with 300 indoor and 300 outdoor images, totaling 600 labeled images.

2) Evaluation method: The most used evaluation method for fiducial points detection is the square root of the mean squared error between the computed points and the ground truth, normalized by the interocular distance [24]. Denoting, respectively, the computed points and the ground truth as $\boldsymbol{l}^f = [x_1^f, y_1^f, \ldots, x_N^f, y_N^f]^T$ and $\boldsymbol{l}^g = [x_1^g, y_1^g, \ldots, x_N^g, y_N^g]^T$, the error is given by

$$\text{NRMSE} = \frac{\sum_{i=1}^{N}\sqrt{(x_i^f - x_i^g)^2 + (y_i^f - y_i^g)^2}}{d_{\text{io}}N}, \qquad (3)$$

where d_{io} is the interocular distance.

3) Two-layer neural network: First, an MLP is build, called here *BaseNet*, consisting of an hidden layer with 300 neurons and ReLU [17] activation. The inputs are the gray scale images stacked into an array ($\boldsymbol{x} \in \mathbb{R}^{256*256*1}$), totaling 19M parameters. The training used *stochastic gradient descent* (SGD) algorithm with a learning rate of 0.01 and a batch of size 32.

As discussed in Section III-B.2, a MLP neural network has a large number of parameters due to high dimensional inputs. As can be noticed in Fig. 4, although the network seems to learn something, it does not generalize for being incapable of extracting sufficient information for the image pixels. Therefore, the network predicts a pattern mask to minimize the mean error, as depicted in the first line of Fig. 5.

4) Convolutional neural network: In the second experiment, a ConvNet was assembled keeping the same amount of hyperparameters to have a fair performance comparison. The topology consists of 4 blocks of the type CONV→ReLU→*MaxPooling*, keeping the convolutional filters kernel size 3×3, padding of size 1 in all blocks and max-pooling layer using a filter of size 2×2. The number of filters in each block are 32, 64, 128, 256, respectively. There

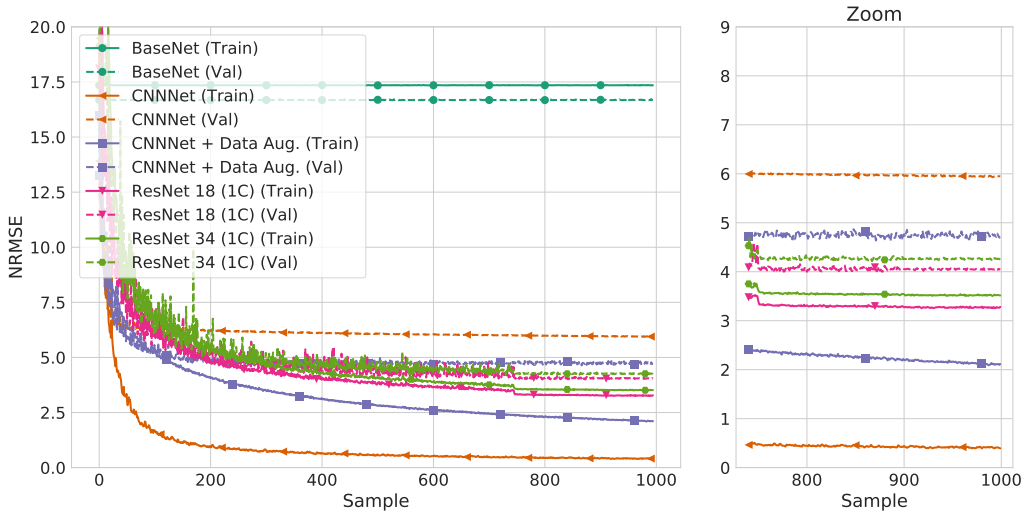

Fig. 4. Results NRMSE over iteration time for networks used in fiducial points extraction task.

are two-layer Fully-Connected (FC) at the end to generate the output. The inputs are 256×256 pixels grayscale images. The training was realized using SGD algorithm with a learning rate of 0.01.

Better results were obtained using this topology with the same parameter numbers as BaseNet, but having convolutional and pooling layers (0.40613 to 0.19053, as can be seen in TABLE I). However, as one can note in Fig. 4, the high number of parameters resulted in a network with high variance. To deal with this problem, a data augmentation scheme (CNNNet + Aug in TABLE I) was employed. The new augmented data contains small modifications of the image in training set. The augmented data contains horizontal flip and random crop of the image with size 224×224 pixels. This strategy led to improvement since the variance was lowered and the error is reduced to 0.12848.

5) ResNets: Deeper networks can extract more useful information from an image. The model used in this work is based on the same topology designed to classify images in the ImageNet database [17]. However, since the fiducial points database is much smaller than ImageNet dataset, the number of filters in each layer was reduced to half. So, there are fewer parameters, a better generalization capability and increase in training speed if compared to the network designed for the ImageNet task.

Two versions of this topology were evaluated, ResNet 18 (18 layers) and ResNet 34 (34 layers), both using grayscale images as input. A learning rate of 0.1 was employed to ResNet 18 while a rate of 0.05 to ResNet 34. Both use the same *data augmentation* strategy from the previous topology.

As can be noticed in TABLE I, there is an increase in the results (error of 0.12103 to ResNet 18). Although it is a small improvement, it is worth to mention that this network has only 700k parameters making the training much faster and more efficient. In Fig. 5, one can note that ResNet 18 could provide a fine tuning in the cheeks as well as could better align mouth points with unusual positions (5th column). The results for ResNet 34 was worse than for ResNet 18, probably because of the increase in the number of hyperparameters which can lead to training instability. Finally, due to speed, ResNet 18 was chosen to recognize people in this work.

TABLE I
COMPARISON BETWEEN EVALUATED ARCHITECTURES.

Topology	**#Params.**	**NRMSE**		
		Common	Challenging	300-W
BaseNet	19M	0.22397	0.40613	0.28672
CNNNet	17M	0.08050	0.19053	0.13150
CNNet + Aug.	17M	0.06610	0.12848	0.09753
ResNet 18	700k	0.05641	0.12103	0.08752
ResNet 34	700k	0.05745	0.13351	0.09146

B. *Facial Recognition*

Aiming recognize known people, we built a dataset with 20 images from 5 people (Fig. 6) from Signals Multimedia and Telecommunications (SMT) laboratory at UFRJ. As can be noticed, there is no control in pose, illumination or image quality. Each image passed through the process of face extraction, facial features extraction, alignment, and embedding, as described in Section III. Then, these images were used to train an SVM classifier with linear kernel and regularizing factor of 1.3, resulting in an accuracy of 0.95 ± 0.13 (using *k-fold* with $k = 10$).

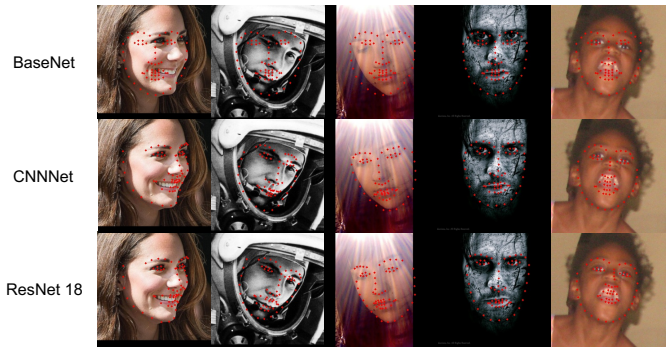

Fig. 5. Samples from test database with estimated fiducial points superimposed.

Fig. 6. Some samples of the dataset built.

All code is developed in Python, using `Pytorch`, `sklearn`, `dlib` and `opencv2` libraries. It is encapsulated in classes and modularized to ease of use and is available at `http://github.com/wesleylp/CPE775`, under MIT license.

Without using any production framework (e.g., CNTK, MxNet, and Caffe2), code optimization nor algorithm compression on the built networks, the system reached 23.23 ± 0.60 fps rate using a computer Intel Core i7-6850k, 32GB RAM memory and a video card GTX 1080. Without using GPU, the rate is 18.99 ± 0.68 fps. Fig. 7 shows the system being used through a webcam.

V. Conclusion

This work describes a modular system capable of recognizing people. It has five steps: face segmentation, facial features detection, face alignment, embedding, and classification. Deep learning was employed to locate facial fiducial points (or facial features) and embedding. The best results were obtained using a customized architecture of residual network with 18 layers, drastically reducing the number of parameters from 19M to 700k. Due to the employed preprocessing (face detection, fiducial points detection, face alignment and embedding), the classification model is simple but efficient. The system has competitive accuracy (0.95 ± 0.13), runs in real-time (23.23 ± 0.60 fps) and is fully available under MIT license.

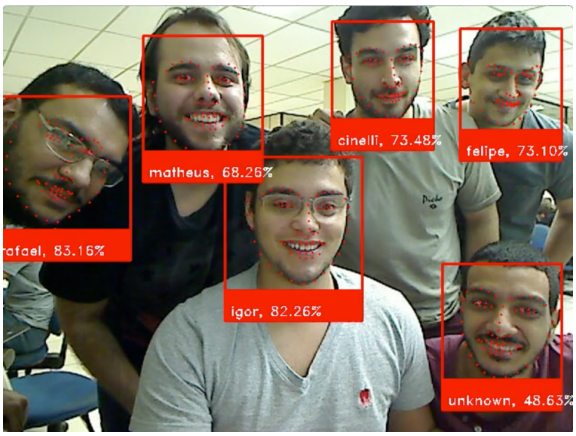

Fig. 7. System working in real-time through a webcam.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, nov 1998.

[3] J. Sivic, M. Everingham, and A. Zisserman, ""who are you?" - Learning Person Specific Classifiers from Video," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Miami, FL, USA: IEEE, jun 2009, pp. 1063–6919.

[4] L. Wolf, T. Hassner, and I. Maoz, "Face Recognition in Unconstrained Videos with Matched Background Similarity," in *Proc. of the IEEE Conf. on Computer Vision Pattern Recognition (CVPR)*. Providence, RI, USA: IEEE, jun 2011, pp. 529–534.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, ""Microsoft COCO: Common Objects in Context," in *Proc. of the 13th European Conf. on Computer Vision (ECCV)*. Zurich, Switzerland: Springer, sep 2014, pp. 740–755.

[6] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Fisher Vector Faces in the Wild," in *BMVC*, sep 2013, p. 4.

[7] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Columbus, OH, USA: IEEE, jun 2014, pp. 1701–1708.

[8] ——, "Web-scale Training for Face Identification," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, jun 2015, pp. 2746–2754.

[9] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, jun 2015, pp. 815–823.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, jun 2015, pp. 1–9.

[11] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *Proc. of the Int. Conf. on Learning Representation (ICLR)*, Banff, Canada, apr 2014.

[12] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," *CMU School of Computer Science*, 2016.

[13] P. Viola and M. J. Jones, "Robust Real-Time Face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, may 2004.

[14] X. Zhu and D. Ramanan, "Face Detection, Pose Estimation, and Landmark Localization in the Wild," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Providence, RI, USA: IEEE, jun 2012, pp. 2879–2886.

[15] D. E. King, "Max-Margin Object Detection," *arXiv preprint arXiv:1502.00046*, 2015.

[16] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. San Diego, CA, USA: IEEE, jun 2005, pp. 886–893.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, USA, dec 2012, pp. 1097–1105.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, jun 2014.

[19] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. of the Int. Conf. on Machine Learning (ICML)*, Lille, France, jul 2015, pp. 448–456.

[20] M. Lin, Q. Chen, and S. Yan, "Network in Network," in *Proc. of the Int. Conf. on Learning Representation (ICLR)*, Banff, Canada, apr 2014, pp. 1–10.

[21] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 Faces in-the-Wild Challenge: The First Facial Landmark Localization Challenge," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Sydney, NSW, Australia: IEEE, dec 2013, pp. 397–403.

[22] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing Parts of Faces Using a Consensus of Exemplars," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Providence, RI, USA: IEEE, jun 2011, pp. 545–552.

[23] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *European Conference on Computer Vision*, 2012, pp. 679–692.

[24] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz, "Robust Face Detection Using the Hausdorff Distance," in *Proc. of the Int. Conf. on Audio- and Video-Based Biometric Person Authentication*. Halmstad, Sweden: Springer, jun 2001, pp. 90–95.