Matthew Bock, Andrew Guinn, Jose Prado, Marco Tando

HACS498

Automated Video Facial Recognition Literature Review

Techniques for Facial Recognition for Video Footage

In general, the procedure to engage in Automated Video Facial Recognition (hereafter referred to as AVFR) is as follows: first, facial detection must be done to determine if, within a given frame, there exists a face, second, the face must be aligned so that factors such as pose and size are controlled for, third, features must be extracted from the face that has been identified, and fourth and finally, the features exacted from the image must be matched to known faces of interest [20]. There are broadly, two main approaches for AVFR: conventional, and deep learning methods [1], although these approaches can also be subdivided into many different types such as Statistical and Geometry Feature [20]. Conventional methods require hand-crafted feature extraction and pre-trained classifiers, while deep learning strategies can automatically learn features and classifiers together but require large amounts of data [1]. Conventional methods are typically modeled as individual specific face detectors which allows a system to easily add or subtract individuals [1]. Ensembles of conventional classifiers can be used to improve the classification strength of a system in scenarios of imbalanced data [1]. Conventional methods are generally found to perform worse on AVFR tasks than deep learning strategies [1][19]. Deep learning strategies can model non-linear and discriminative feature representations (such as Gabor features [19]), and therefore can have high accuracy but are more computationally expensive and require large amounts of data to train [1]. In general, there is a tradeoff between algorithm accuracy and algorithm efficiency [1], but one cannot carelessly just increase the algorithm complexity to improve efficiency, as adding more complexity to, say, a neural network can actually harm accuracy if done ungracefully [25].

Linear/Conventional Methods

Eigenfaces

In this architecture, the feature space of the image matrices is mapped to a lower dimensional space using Principal Component Analysis [15]. A scatter matrix is generated based upon the sample image(s) and the mean image of the sample set and a projection matrix is determined that maximizes the determinant of the matrix [15]. It seems to be the case that this projection matrix is used to determine whether the pictured face is of a person of interest or not [15]. There are methods of paralleling the operation of Principal Component Analysis through the establishment of a message passing interface in order to improve its training time [16]. Of the methods explored in [15], this was generally the least accurate but had more consistent performance with differing levels of noise than Local Binary Patterns [15].

Fisherfaces

This architecture is a linear classifier that was developed in response to the issue of lighting conditions with regards to facial recognition [15]. This architecture functions similarly to the Eigenfaces architecture except uses a class specific linear in order to reduce the dimensionality of the image and then a classifier to further reduce the feature space [15]. Of the methods explored in [15], this was the second most accurate but had more consistent performance with differing levels of noise than Local Binary Patterns [15].

Local Binary Patterns

This architecture operates by converting an image into a series of local regions of three pixels by three pixels where each pixel is assigned a numeric value based on characteristics of the pixel [15]. The middle pixel is set as the threshold, and each pixel is then assigned 1 if its value exceeds the threshold or 0 otherwise [15]. The eight binary values are converted into one decimal value which is placed into a histogram, which is then compared to known profiles to determine a match [15]. Of the methods explored in [15], this was generally most accurate across multiple light exposures and video resolutions, however, its performance rapidly degraded given more exposure to noise [15].

Deep Learning Methods

Deep Learning architectures are divided into two main architectural types: multi-input networks and multi-task learning networks [19]. The former type of architectures generate multiple images of different patches or poses and involve multiple networks that examine certain features in a “one-to-many” approach [19]. The latter kind of architectures use side tasks, such as pose, illumination, and expression estimations, in order to solve the main task, in this case, classification [19]. Usually, the lower layers of the network are shared among the tasks which then branch off into task specific outputs which are used to determine the classification [19].

Cross-Correlation Matching CNN

This CNN architecture utilizes pair-wise cross-correlation matching along with facial representations learned through triple-loss optimization [1]. This works by training the model with representations of triplets: one “anchor” image, one positive image of the subject, and one negative image of some other subject. Triplet loss functions generate a more suitable representation of the embedding space than other loss functions, which improves the effectiveness of the training process: instead of representing all faces as a single point in the embedding space, instead, all faces from one person exist such that there is simply a margin enforced between them and other faces [1] [5]. This can have some representation benefits, but the method requires careful sampling [1]. The use of a triplet-loss function for learning may prevent the use of simple distance measures such as euclidean or cosine [1]. The architecture has three main components: feature extraction, cross correlation matching, and triplet-loss optimization [1]. The feature extraction, as the name implies, extracts feature maps for corresponding features of different images. The cross-correlation matching utilizes a matrix Hadamard product followed by a fully connected layer that simulates adaptive weighted cross-correlation in order to calculate the likelihood that two photos are of the same person based on the extracted feature maps [1]. This method will typically utilize synthetic face generation to expand the testing set. One advantage of this architecture is that fewer parameters are required for training to avoid over-fitting when compared to other networks of a similar size [1]. Of the methods explored in [1], this is one of the least accurate but one of the fastest computationally.

Trunk-Branch Ensemble CNN

This CNN architecture has one “trunk” network which branches off into several “branch” networks which focus their processing efforts on certain components of an image [1]. This architecture allows for some information to be embedded into the processing of each of the branches, which allows for reducing the cost of computation compared to multiple separate networks [1]. This architecture uses the Mean Distance Regularized triplet loss function, which regularizes the output of traditional triplet loss functions [1]. Of the methods explored in [1], this architecture is one of the most accurate but among the most computational complex.

HaarNet

HaarNet is an instantiation of the Trunk-Based Ensemble with some refinements. The trunk focuses on a general perspective of the face while the branches of the network process specific features of the image in a Haar fashion [1]. This architecture also uses a second order regularized triplet-loss function in order to learn more distinctive representations for subjects with similar faces [1]. This function overcomes the normal triplet-loss functions' issues with simple distance measures [1]. Additionally, a fine-tuning stage is placed in the network in order to embed the correlation of facial regions of enrollment for individuals of interest to increase the accuracy of the model [1]. Of the methods explored in [1], this architecture is the most accurate but among the most computational complex.

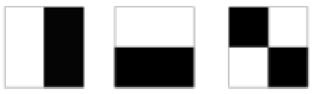
Fig. 7: Haar-like features used in branch networks [25].

Deep CNNs Using Autoencoder

This architecture is built upon supervised autoencoders that attempt to represent the divergence between still images and video (still images in this case being how the target individuals are encoded) [1]. The autoencoder uses weighted, pixel-wise functions to construct clear regions of interest or discriminative face embeddings, which can then be passed to a classification network to reach a conclusion [1]. Of the methods explored in [1], this is one of the least accurate but one of the fastest computationally.

ResNet

This architecture is one of the most pervasively used deep learning architectures in recent deep learning systems for AVFR [19]. The ResNet architecture is a type of Convulutional Neural Network, but innovates by having layers that, instead of modeling a function H(x), model a residual mapping F(x) = H(x) – x, and just add x back in to get F(x) [25]. This can be done in a feed-forward, Convolutional Neural Network by having shortcut connections that skip layers and are added to the output of layers that are skipped [25]. In practice, this means the layers are trying to approximate an identity mapping, which is closer to optimal in the case of image processing than a zero mapping [25]. This does not add extra parameters or complexity to existing algorithms and ResNets can be trained in the usual method as regular Convolutional Neural Networks (such as with Stochastic Gradient Decent) [25]. This strategy reduces the effect of model degradation when adding more layers to the network which allows for easier optimization and greater accuracy than other architectures [25]. This architecture forms the basis of several of the most accurate deep learning AVFR methods [19]. However, in a direct comparison, GoogleNet tends to perform with higher accuracy at lower number of operations [26].

FaceNet

This architecture, developed by engineers at Google, is based upon a Convolutional Neural Network called Inception (for more information, look at [23]), which is trained using Stochastic Gradient Descent and generates results that are L2 normalized to create a facial embeddings which are used to identify the subject in an image [5]. This method utilizes a triplet loss function [5]. This network at the core of the architecture was tested in several different configurations that manipulated the detail of the image that was processed as well as the number of layers used in the network. The result of this architecture was a 95% accuracy of the Youtube Faces DB and a 99% accuracy on LFW dataset, which outperformed state of the art at the time [5]. Implementations of the FaceNet algorithm are publicly available [6].

OpenFace

The OpenFace architecture is based upon the FaceNet architecture [17]. In the training stage, a Convolutional Neural Network is trained in a supervised manner using a triplet loss function, which results in a model which can extract features from images [17]. During the classification stage, images are passed to a face detector from dlib, which is a C++ library which implements a variety of deep learning algorithms and constructs [17]. If a face is identified, a preprocessing step occurs in which the identified face is subject to an affine transformation which centers the face in the image [17]. Finally, the image is processed by the neural network which outputs a representation of the face which is classified via sklearn's SVM algorithm [17]. OpenFace can utilize CUDA functions which can speed up processing [17]. Compared to the OpenCV implementation of EigenFaces, Fisherfaces, and Local Binary Patterns, OpenFace has a superior performance, however, performs worse than DeepFace (from Facebook) and FaceNet (from Google) [17]. The code for Openface is online [11].

DeepFace

DeepFace is an architecture, developed by engineers at Facebook, that is based upon the Alexnet image recognition network (for more information regarding the architecture, view [24]) [18][19]. The main differentiating feature of the DeepFace architecture is to generate a 3D representation of faces in images to then extract features for processing in the neural network [18]. As faces are 3D objects, it was hypothesized that a 3D representation was appropriate in properly analyzing the features of the face [18]. The architecture outperforms the OpenFace architecture but does not outperform FaceNet [17].

ArcFace

Arcface is not a distinct neural network architecture, rather is a novel loss function [21] (in fact, most of the architectures used to demonstrate the utility of ArcFace are Resnet architectures[25]). Arcface is an Additive Angular Margin loss function [21]. This is a variant of the Additive Cosine margin, which both are descended from Multiplicative Angular Margin [21]. An Angular Margin gives a different way to represent the borders between different classes [21]. Additive Cosine margin has three advantages over Multiplicative Angular Loss: (1) easy to implement with few hyper-parameters; (2) does not rely on Softmax supervision to converge; (3) better performance [21]. Additive Angular Margin in turn is an improvement over Additive Cosine margin because it achieves a more accurate convergence [21]. This loss function performs well on many different network architectures [21], and in the most optimal configuration outperforms all other loss function/deep learning architectures surveyed [19].

$$L_7 = -\frac{1}{m}\sum_{i=1}^{m} \log \frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}},$$

ArcFace

Range Loss

Range Loss is a novel loss function implemented on the VGG neural network architecture [29]. Range Loss was formulated to address the issue of long-tailed distributions that caused issues for other loss functions: distributions of samples which have a large number of occurrences far from the center of the distribution, which seems to be the case with many facial image distributions given the large variation in lighting, pose, and more [29]. The Range Loss Function will calculate gradients and do backpropagation based on the overall distance between classes within one mini-batch instead of over pairs [29]. This algorithm outperforms DeepFace, but is outperformed by Arcface and CoCo[19].

$$\mathcal{L} = \mathcal{L}_M + \lambda\mathcal{L}_R = -\sum_{i=1}^{M} \log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} + \lambda\mathcal{L}_R$$

Range Loss

CoCo Loss

The CoCo loss is a novel loss function implemented on an unspecified neural network architecture [29]. The CoCo loss function draws from properties of the softmax function and cosine distances: the function is able to effectively discriminate between features in higher dimensional space and maintain a class centroid while also minimizing the intra-class distance [29]. This loss function led to the best performance of all the listed loss functions in the paper [19].

$$\mathcal{L}^{COCO}(\boldsymbol{f}^{(i)}, \boldsymbol{c}_k) = -\sum_{i\in\mathcal{B},k} t_k^{(i)} \log p_k^{(i)} = -\sum_{i\in\mathcal{B}} \log p_{l_i}^{(i)},$$

CoCo Loss

Challenges for Automated Video Facial Detection

Model Development

In general, AFVR strategies will suffer from issues with acquiring sufficient data to train automated models and then to identify a particular subject. In order to address the first problem, strategies have been used such as multiple face representations, synthetic face generation, and using data from other people [1]. These strategies increase the robustness of the model by expanding the set of images that is used to train the model. Academic research communities have generally tried to make up for the lack of large face data sets by carefully designing loss functions that can train networks efficiently with a smaller training sample [19].

One important consideration in the development of a model is that although the task of identifying faces in images is similar to the task of identifying faces in videos, there are many qualities that are distinct to video which means that special considerations need to be taken when training a model for the purpose of video facial recognition [13]. For instance, if images are used to aid in the training of the algorithm, it is useful to blur the images to simulate the quality of video representations of faces [13].

One consideration when developing a model for classification is the choice of loss function. Loss functions can influence how effectively a model is developed for learning given a data set. There are several major classes of loss functions that have been used in the area of AVFR: Euclidean distance, Angular/cosine margin, and Softmax [19]. Euclidean distance based loss functions are based upon embedding the features of the image into Euclidean space to calculate the dimensions of features [19]. This involves compressing inter-variance and enlarging intra-variance

[19]. Contrastive, triplet, and center loss are examples of Euclidian algorithms [19]. The former two algorithms tend to suffer instability based upon he selection of training data, while the latter imposes significant memory requirements on GPUs [19]. Angular/cosine margine functions use angular distances in order to separate learned features in the feature space [19]. Angular/cosine loss functions include L-Softmax and A-Softmax, CosineFace, and ArcFace [19]. Angular/cosine-margin-based loss explicitly adds discriminative constraints on a hypershpere manifold which match the prior human face, which allows angular/cosine margin functions to perform better result on clean data than a Euclidean-distance function, but are vulnerable to noise which makes them generally perform worse than center loss or softmax functions [19]. Softmax loss functions use variations of the softmax function to calculate loss [19]. Examples of this type of loss function include the L2-softmax, which also uses L2-normalization, and Ring Loss, which encourages the norm of samples though a learned parameter [19].

$$L_1 = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}},$$

Softmax Function

Video Acquisition and Processing

Environmental conditions such as ambient lighting, blur, and occlusion can all cause distortion to facial images captured by video footage [1][13]. The position and pose of the subject relative to the position and pose of the subject in reference examples can have a large impact on the success of models [1][13][19]. Additionally, the effectiveness of machine learning models can be very dependent on certain characteristics of subjects such as race [2][13]. Likely because of the influence of factors such of these, in real world trials, the overall effectiveness of automated facial recognition has been mixed, with notable failures in the UK but successes in China [3][4].

There are some ways to mitigate some of the issues referenced above. For instance, the issues regarding ambient light could be mitigated through the use of thermal imagery in addition to or in lieu of optical imagery, which has already seen effectiveness in this regard [7]. Other sensors could be used to augment the accuracy of classifications in inclement situations [13].

Although likely not a critical concern for the purposes of this particular development task, it is important to consider adversarial situations and how our architecture can be designed with adversary mitigation in mind. Deep neural network architectures in particular are vulnerable to adversarial data distortions which can trigger misclassifications although some architectures, such as OpenFace, are more vulnerable than others [10]. In order to address adversarial scenarios, one must either utilize techniques that make the network more robust to noise during the training phase or by using preprocessing algorithms, such as selective dropout, to filter out noise from ingested data [10].

Conclusions

Deep-learning algorithms tend to outperform more traditional machine learning algorithms on AVFR tasks [1][19]. There seems to be a strong correlation between the complexity of a model and its accuracy [1][5]. The source code for several of the facial recognition algorithms exists online [6][8][11][12][22], while some free services exist that provide facial recognition functionality [9][14]. Although accuracy metrics are extensively studied for many of the algorithms [1][3][5][13][15][16][18][19][26], the study of the classification and training time is not commonly evaluated [1][26]. This combined with the fact that the sensitivity to noise of algorithms has not been extensively studied [10][19] suggests that testing of these algorithms is required in order to determine the most suitable algorithm for our use-case. Luckily, large scale datasets are publicly available for use to train the algorithm, so that should not present an issue [27][28].

Citations

[1] S Bashbaghi, E Granger, R Sabourin, M Parchami, Deep Learning Architectures for Face Recognition in Video Surveillance, Deep Learning in Object Detection and Recognition, Springer, 2018.

[2] Lohr, Steve. “Facial Recognition Is Accurate, If You're a White Guy.” *The New York Times*, The New York Times, 9 Feb. 2018, www.nytimes.com/2018/02/09/technology/facial-recognition-race-artificial-intelligence.html.

[3] Dodd, Vikram. “UK Police Use of Facial Recognition Technology a Failure, Says Report.” *The Guardian*, Guardian News and Media, 14 May 2018, www.theguardian.com/uk-news/2018/may/15/uk-police-use-of-facial-recognition-technology-failure.

[4] Charette, Robert N. “Automated Facial Recognition: Menace, Farce, or Both?” *IEEE Spectrum: Technology, Engineering, and Science News*, IEEE Spectrum, 29 May 2018, spectrum.ieee.org/riskfactor/computing/it/automated-facial-recognition-menace-farce-or-both.

[5] Schroff, et al. “FaceNet: A Unified Embedding for Face Recognition and Clustering.” *[Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field*, 17 June 2015, arxiv.org/abs/1503.03832.

[6] Samberg, David. “FaceNet.” *Github*, 096ed770f163957c1e56efa7feeb194773920f6e, Google , 20 Apr. 2018, github.com/davidsandberg/facenet.

[7] U.S. Army Research Laboratory. "Face recognition technology that works in the dark." ScienceDaily. ScienceDaily, 16 April 2018. <www.sciencedaily.com/releases/2018/04/180416142443.htm>.

[8] Kumar, Varun. “15 Efficient Face Recognition Algorithms And Techniques.” *RankRed*, 3 Nov. 2017, www.rankred.com/face-recognition-algorithms-techniques/.

[9] “Face API - Facial Recognition Software | Microsoft Azure.” *A Beginner's Guide | Microsoft Azure*, Microsoft, azure.microsoft.com/en-us/services/cognitive-services/face/.

[10] *Goswami, Gaurav et al. “Unravelling Robustness of Deep Learning based Face Recognition Against Adversarial Attacks.” CoRR abs/1803.00401 (2018): n. pag.*

[11] Amos, Brandon, et al. “OpenFace.” *GitHub*, c2d3b2df055ae8637eff28422d7916c1575a6e83, GitHub, github.com/cmusatyalab/openface.

[12] Ageitgey. “face_recognition.” *GitHub*, b8fed6f3c0ad5ab2dab72d6251c60843cad71386, GitHub, github.com/ageitgey/face_recognition

[13] Bajić, Milan & Debevc, Matjaž. (2017). Face recognition - machine learning highway to efficiency. Polytechnic and design. 5. 12-17. 10.19279/TVZ.PD.2017-5-1-02.

[14] "Amazon Rekognition – Video and Image - AWS." *Amazon*, Amazon, aws.amazon.com/rekognition/.

[15] Jaturawat, Phichaya, and Manop Phankokkruad. "An Evaluation of Face Recognition Algorithms and Accuracy Based on Video in Unconstrained Factors." *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Nov. 2016, pp. 240–245., doi:10.1109/iccsce.2016.7893578.

[16] Shouman, Dalia, and Salma Hamdy. "Parallel Architecture for Face Recognition Using MPI." *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 1, 1 Nov. 2017, pp. 425–430., doi:10.14569/ijacsa.2017.080154.

[17] Amos, Brandon et al. "OpenFace : A general-purpose face recognition library with mobile applications." (2016).

[18] Taigman, Yaniv, et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification." *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, doi:10.1109/cvpr.2014.220.

[19] Wang, Mei, and Weihong Deng. "Deep Face Recognition: A Survey." *ArXiv*, 28 Sept. 2018, doi:arXiv:1804.06655v7.

[20] Patil, Shailaja A, and P J Deore. "Face Recognition: A Survey." *Informatics Engineering, an International Journa*, vol. 1, no. 1, Dec. 2013, pp. 31–41., airccse.org/journal/ieij/papers/1113ieij05.pdf.

[21] Deng, Jiankang, et al. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition." *ArXiv*, 23 Jan. 2018, doi:arXiv:1801.07698v1.

[22] Guo, Jia, and Jiankang Deng. "Insightface." *Github*, afa3ad8892d7a3967a0caff90e978873e5ddd146, Github, github.com/deepinsight/insightface.

[23] Szegedy, Christian, et al. "Going Deeper with Convolutions." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, doi:10.1109/cvpr.2015.7298594.

[24] Krizhevsky, Alex, et al. "ImageNet Classification with Deep Convolutional Neural Networks." *Neural Information Processing Systems*, 2012, pp. 1097–1105., papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[25] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, doi:10.1109/cvpr.2016.90.

[26] Canziani, Alfredo, et al. "An Analysis of Deep Neural Networks for Practical Applications." *ArXiv*, doi:arXiv:1605.07678v4.

[27] "MS-Celeb-1M: Challenge of Recognizing One Million Celebrities in the Real World - Microsoft Research." *Software Asset Management – Microsoft SAM*, Microsoft, 1 Apr. 2016, www.microsoft.com/en-us/research/project/ms-celeb-1m-challenge-recognizing-one-million-celebrities-real-world/.

[28] Grgic, Mislav, and Kresimir Delac. *Face Recognition Homepage - Databases*, University of Zagreb, www.face-rec.org/databases/.

[29] Zhang, Xiao, et al. “Range Loss for Deep Face Recognition with Long-Tailed Training Data.” *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, doi:10.1109/iccv.2017.578.

[30] Zhang, Xiao, et al. “COCO Loss.” *Github*, 4051c62817756b566911fc7f1154beda87b0ef72, Github, github.com/sciencefans/coco_loss.