Literature Survey

Face Verification Problem on the Large Scale Datasets

Ph.D. candidate: **Artem Komarichev**

Research advisor: **Prof. Xue-wen Chen**

Department of Computer Science

Wayne State University

USA

1 Introduction

In the recent few years the interest in face recognition problems was increased significantly. One of the reasons behind this that now we have more powerful machines than couple decades ago which we can use to run more complex models. Moreover with the rapid grough in the development of convolutional netural networks (CNN) and deep architectures we could learn much better feature representations to make our predictions more accurate. Plus, there are a plenty of potential areas where we can apply our models such as: biometrics, information security, law enforcement and surveillance, access control and many others [1].

Face recognition can be divided into several directions:

- face verification
- face identification
- face detection
- face clustering

The main goal of *face verification* is to define whether two face images belong to the same person or not. *Face detection* is basically defining are there faces in the image, and if it is true, just return their locations in this digital image [2]. The goal of *face identification* is to assign each image to appropriate class in a dabase [3]. *Face clustering* defines common people among all available face images in the database. I want to concentrate only on face verification problem in this literature survey.

What does make face recognition problem hard to be solved? Some: illumination, rotation, different poses, occlusion [4], facial expressions and so on. One of the datasets which reflects these difficulties is PIE (pose, illumination, and expression) dataset [5]. The examples of image pairs you can see in Fig. 1 (I will add image next time). And it was considered to be very difficult to solve until the guys from Google proposed their solution [6] and solved it.

The other issue which I would like to consider here is using of large-scale datasets in solving problem. In order to effectively run any face verfification model on huge dataset we most likely need to parallelize our solution. There is two possible ways to make this: model parallelization and data parallelization [7]. I will cover them in more details later and show some state-of-the-art applications.

I will also descibe two widely used datasets for evaluation: Labeled Faces in the Wild (LFW) and Youtube Faces DB (YTF).

2 DeepFace

2.1 Face alignment

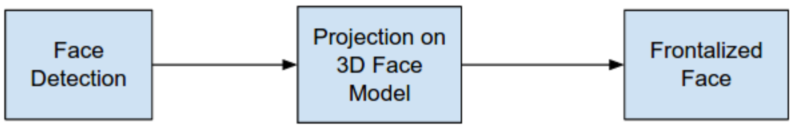

Figure ?.?: Frontalization steps.

2.2 DeepFace architecture

The main reason behind using feature representations learned through DNN instead of using handcrafted engineered features because they shows better prediction accuracy. The more data available, the better features could be extracted using deep architectures.

The big picture of DeepFace model is represented below.

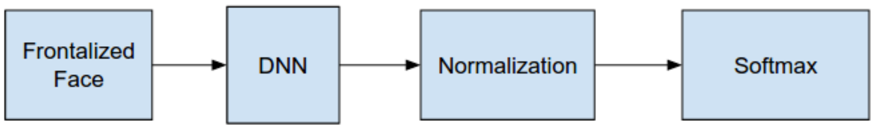

Figure ?.?: DeepFace architecture.

Input signal to the DNN is a 3D-aligned faces in RGB. The size of the input image is 152×152 pixels. The process of getting 3D-aligned face images was described in the previous section.

Speaking of DNN architecture, it has two convolutional layers (C1@$11 \times 11 \times 32$ and C3@$9 \times 9 \times 16$), one max-pooling layer (M2:$3 \times 3, stride = 2$), three locally connected layers (L4@$9 \times 9 \times 16$, L5@$7 \times 7 \times 16$, and L6@$5 \times 5 \times 16$) and final two fully connected layers (F7:4096d and F8:4030d). The key difference between convolutional layer and locally connected (LC) one that there is no weights sharing across the whole image. Each patch in LC layer has his own filter. In contrast to LC layers, convolutional layers has one filter for every patch extracted from image. Having more parameters to train in the LC layers doesn't lead to an increase in computations. The only reason why they used this type of layers because they have pretty huge training dataset which made it possible to train them. The number of parameters was close to 120M, where 95% was from LC and FC layers. The last FC layer represents softmax classifier, where the number of classes is equal to 4030 unique identities in the training dataset.

As loss function was chosen cross-entropy loss, becuase the goal is to minimize the probability of the false classifications. For example, if p_i is a prediction that this image belongs to class i, than loss function L could be calculated in the following way: $L = \sum_i^N -log(p_i)$, where N is a number of classes. To update parameters in the model was used stochastic gradient descent (SGD). The size of the mini-batch was equal to 128. One of the distinctive property of this DNN that gradients, calculated by standard backpropagation algorithm, are sparse. They mentioned that approximately 75% of them were equal to zero in the last five layers. That happened most likely because they used as non-linear activation function a rectifier linear units: $max(0, x)$. Sparsity was encorouged in the last time. Applying such sparsity techniques as dropout [10], maxout [8] shows better prediction accuracy.

In order to make their DNN more robust to illuminations, they used normalization as a final stage. The basic idea of normalization, that every feature value was divided by the largest value across all these features. This technique gave them distribution of each feature between zero and one.

2.3 Metrics to estimate verification

2.4 Experiments and Results

Comparing different reduced architectures (1) without second conovolutional layer, 2) without first two locally connected layers, and 3) without three these layers) with the original one, they showed that the deep of the architecture is a critical issue. Deeper DNN can learn better feature representations. Therefore, deeper architectures shows higher final accuracy than shallower ones does.

In conclusion, they achieved the new state-of-the-art on the LFW and YTF datasets by the time when the paper was published; they proposed a new alignment techniques based on 3D modeling of the face images in the unconstrained settings; they came up with a new DNN architecture which was trained on their own huge dataset.

3 FaceNet

Google proposed the new idea [6] and showed how face verification problem could be solved on the large-scale dataset. Their solution actually could be applied to different problems such as: face verification, face recognition and clustering. The key idea of their approach that they use Euclidean embedding space to find the similarity or difference between faces. The architecture of their model is represented on Fig. ?.?

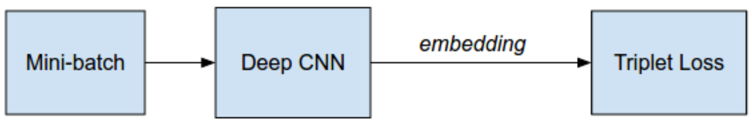

Figure ?.?: FaceNet architecture.

3.1 Triplet Loss

Their most significant contribution is so-called "triplet loss". During the training process this loss minimizes the distances between similar faces and maximizes one between different faces.

Loss function that they were using was:

$$Loss = \sum_{j}^{N} \left(||f(x_j^a) - f(x_j^p)||_2^2 - ||f(x_j^a) - f(x_j^n)||_2^2 + \gamma \right) \qquad (3.1)$$

where N is a possible set of all triplets in the dataset; $f(x)$ - embedding, that translates image x into Euclidean space; x_j^a - "anchor" image; x_j^p - "positive" image, which should be close to an anchor image; x_j^n - "negative" image; γ - margin between positive and negative images.

We are interested in such triplets which violetes the following constraint $||f(x_j^a) - f(x_j^p)||_2^2 + \gamma < ||f(x_j^a) - f(x_j^n)||_2^2$. Only these triplets will contribute to our model during the training process. The only open question how to select such triplets.

They were selecting such pairs where distance between "anchor" and "positive" image was maximized ("hard positive") and the distance between "anchor" and "negative" was as small as possible ("hard negative"), but not zero. It is obviously tricky to look for such triplets in the whole dataset each iteration. Intead they used mini-batch approach to find these "hard positive" and "hard negative" pairs. The size of the batch in their experiments was equal to 1800 examplars.

The problem with selecting hard negatives that $L2$ distance could be equal to 0. Instead they were considering only the cases where the distance of hard positive and anchor less than the distance between hard negative and anchor. They called them "semi-hard" examplars.

3.2 Deep Architectures

They used two different deep architectures to extract features:

- "Convnet model" developed by D. Zeiler and B. Fergus [9] based on Krizhevsky architecture [10]. Model was exteneded by adding 1×1 convolutional layer before each standard convolutional layer. These 1×1 convolutional kernels were suggested by [11]. Total number of layers were equal to 22. Number of parameters to train is 140M.
- The other model was inspired by "GoogLeNet" Inception convolutional neural network [12]. Their model is almost the same except that they used L_2 pooling instead of maxpooling in inception layers. Total number of layers were equal to 16. Number of parameters to train is around 6.6M-7.5M.

Inception model could be run on the mobile devices because of the small number of parameters to be trained.

They used rectified linear units in both these models.

3.3 Experiments and results

First I would like to mention that the size of their training dataset is close to 200M faces with 8M identities [6], as stated in Table **??**. It is the biggest training dataset which exists at this moment in industry.

During their experiments they found out that 128 dimension of embedding is good enough for face recognition tasks. It is even possible to use less dimension, 64 for example, without significant loss of accuracy.

Both deep architectures showed almost the same results. Except the fact that Inception model is a way better in terms of performance.

The most interesting part is how the size of training dataset affects accuracy prediction. They compared the following sizes: millions of images, tens of millions, and hundreds of millions. The most significant impovement was got on the training dataset with tens of millions of examplars. So in their experiments they showed that the question how large your training dataset is important. The larger dataset the more accurate model you will get. But to get all these results they have to run their model for almost 700 hours (1 month).

Speaking of the accuracy of their model on two academic dataset: LFW and Youtube Faces DB, they achieved current state-of-the-art results. See results in Table 3.1.

Models	Datasets	
	LFW	Youtube Faces DB
FaceNet	**99.63%**$\pm$0.15	**95.12%**$\pm$0.39
DeepFace	...	...
Parkhi's approach	**98.65%**	**97.3%**
DeepID2	...	...
DeepID2+	...	...
DeepID3	...	...
...	...	...

Table 3.1: State-of-the-art in face verification.

4 Parkhi's deep face approach

Inspired by papers [13, 6], guys from University of Oxford decided to create their own comparably large face dataset [14]. And the main reason behind this that these huge datasets from Facebook and Google are not publically available. This newly collected dataset includes 2.6M images belonging to 2,622 unique identities. It is the biggest training dataset for face recognition in the academia at this moment.

Besides providing algorithm how to collect this new dataset, they also tested three different models on this dataset from [15]. Using their pretrained model they achieved comparably good results on LFW and YTF to the current state-of-the-art.

In terms of face verification problem, they applied the same triplet loss as was suggested here [6]. For embedding function, was chosen affine projection on Euclidean space.

4.1 Model architectures

The architectures A, B and D are same as in [15]. All these three models are same in terms of the type of layers which were used. But they are different in terms of depth. For example, A had 8 convolutional layers and 3 fully-connected (FC) layers. Number of conovlutional layers in B and D models were increased by 2 and 5 respectively.

Input images in all these three models had a size of 224×224 pixels.

Speaking of the number of filters, they started from 64 and after each max-pooling ($2 \times 2; stride = 2$) they increased that number twice, until it reached 512. Max-pooling was not applied ater each convolutional layer only at specific places.

The main important part of these deep architectures is that very small convolutional kernels (3×3) were used across all convolutional layers. By using these small filters they achieved two goals [15]. First, they reduced the number of parameters that should to be trained. Second improvement they achieved could be extracted from the following fact. 5 by 5 filters, for example, can be represented by the stack of two 3×3 consecutive filters without pooling in between. 7×7 filters can be replaced by a stack of three 3×3 filters. This fact shows us that more disciminative decision function can be learned by replacing one non-linear rectifier units by several ones.

The Local Response Normalization (LCN) [10] was not used. Because it was shown that LCN does not give any significant improvements in terms of accuracy of their models.

4.2 Experiments and results

The evaluation was performed on LFW and YTF datasets, but as the training data they used images from their newly collected dataset. More importantly unique identities from benchmark datasets and new one are different.

During the training proces, models B and D used pretrained parameters of A one instead of starting from scratch.

They considred several aspects in their experiments on LFW dataset. I will only describe here three of them. First, they evaluated the deep of architectures and how it depends on the accuracy. The experiments showed that B model (**97.27**%) gave a slight boost in accuracy over model A (**96.70**%), but model D didn't improve results of model B it performed even worse by giving **96.73**% of accuracy. And one of the potential reason for this, that the number of parameters are different. The other one that they used pre-trained parameters from A to train B and D models. By transfering pre-trained parameters one have to be careful with learning rate and momentum. Also they did not investigated random initialization in their experiments. Second, applying 2D allignment on the test dataset gave improvement in accuracy, but applying allignment to training dataset did not improve model. Third, by applying triplet loss they got **99.13%** accuracy. It is significant improvements over model B without embedding.

To sum it up, they used simpler architecture than was used in DeepID-2,2+,3 [16, 17, 18] and their model was trained on the smaller dataset comparing to DeepFace and FaceNet approaches. Nevertheless, they achieved comparable results, see Table 3.1.

5 Conclusion

The one drawback of all these large-scale approaches that they didn't use parallelization techniques explained earlier.

References

[1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM computing surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.

[2] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 1, pp. 34–58, 2002.

[3] H. J. Seo and P. Milanfar, "Face verification using the lark representation," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 4, pp. 1275–1286, 2011.

[4] H. K. Ekenel and R. Stiefelhagen, "Why is facial occlusion a challenging problem?" in *Advances in Biometrics.* Springer, 2009, pp. 299–308.

[5] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression (pie) database," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on.* IEEE, 2002, pp. 46–51.

[6] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *arXiv preprint arXiv:1503.03832*, 2015.

[7] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1223–1231.

[8] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014.* Springer, 2014, pp. 818–833.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR, abs/1312.4400*, 2013.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014.

[13] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on.* IEEE, 2014, pp. 1701–1708.

[14] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” *Proceedings of the British Machine Vision*, vol. 1, no. 3, p. 6, 2015.

[15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[16] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in *Advances in Neural Information Processing Systems*, 2014, pp. 1988–1996.

[17] Y. Sun, X. Wang, and X. Tang, “Deeply learned face representations are sparse, selective, and robust,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2892–2900.

[18] Y. Sun, D. Liang, X. Wang, and X. Tang, “Deepid3: Face recognition with very deep neural networks,” *arXiv preprint arXiv:1502.00873*, 2015.