Probabilistic Face Embeddings

Yichun Shi and Anil K. Jain
Michigan State University, East Lansing, MI
shiyichu@msu.edu, jain@cse.msu.edu

A. Proofs

A.1. Mutual Likelihood Score

Here we prove Equation (3) in the main paper. For simplicity, we do not need to directly solve the integral. Instead, let us consider an alternative vector $\Delta\mathbf{z} = \mathbf{z}_i - \mathbf{z}_j$, where $\mathbf{z}_i \sim p(\mathbf{z}|\mathbf{x}_i)$, $\mathbf{z}_j \sim p(\mathbf{z}|\mathbf{x}_j)$ and $(\mathbf{x}_i, \mathbf{x}_j)$ are the pair of images we need to compare. Then, $p(\mathbf{z}_i = \mathbf{z}_j)$, *i.e.* Equation (2) in the main paper, is equivalent to the density value of $p(\Delta\mathbf{z} = \mathbf{0})$.

The l^{th} component (dimension) of $\Delta\mathbf{z}$, $\Delta z^{(l)}$, is the subtraction of two Gaussian variables, which means:

$$\Delta z^{(l)} \sim \mathcal{N}(\mu_i^{(l)} - \mu_j^{(l)}, \sigma_i^{2(l)} + \sigma_j^{2(l)}). \tag{10}$$

Therefore, the mutual likelihood score is given by:

$$\begin{aligned} & s(\mathbf{x}_i, \mathbf{x}_j) \\ = & \log p(\mathbf{z}_i = \mathbf{z}_j) \\ = & \log p(\Delta\mathbf{z} = \mathbf{0}) \\ = & \sum_l^D \log p(\Delta z^{(l)} = 0) \\ = & -\frac{1}{2}\sum_{l=1}^D (\frac{(\mu_i^{(l)} - \mu_j^{(l)})^2}{\sigma_i^{2(l)} + \sigma_j^{2(l)}} + \log(\sigma_i^{2(l)} + \sigma_j^{2(l)})) \\ & -\frac{D}{2}\log 2\pi. \end{aligned} \tag{11}$$

Note that directly solving the integral will lead to the same solution.

A.2. Property 1

Let us consider the case that $\sigma_i^{2(l)}$ equals to a constant $c > 0$ for any image $\mathbf{x}_i$ and dimension l. Thus the mutual likelihood score between a pair $(\mathbf{x}_i, \mathbf{x}_j)$ becomes:

$$\begin{aligned} & s(\mathbf{x}_i, \mathbf{x}_j) \\ = & -\frac{1}{2}\sum_{l=1}^D (\frac{(\mu_i^{(l)} - \mu_j^{(l)})^2}{2c} + \log(2c)) - \frac{D}{2}\log 2\pi \\ = & -c_1 \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|^2 - c_2, \end{aligned} \tag{12}$$

where $c_1 = \frac{1}{4c}$ and $c_2 = \frac{D}{2}\log(4\pi c)$ are both constants.

A.3. Representation Fusion

We first prove Equation (5) in the main paper. Assuming all the observations $\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_{n+1}$ are conditionally independent given the latent code z. The posterior distribution is:

$$\begin{aligned} & p(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}) \\ = & \frac{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1})} \\ = & \frac{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n|\mathbf{z})p(\mathbf{x}_{n+1}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1})} \\ = & \frac{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)p(\mathbf{x}_{n+1})}{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1})} \frac{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n|\mathbf{z})p(\mathbf{x}_{n+1}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)p(\mathbf{x}_{n+1})} \\ = & \alpha\frac{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{z})p(\mathbf{x}_{n+1}, \mathbf{z})}{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)p(\mathbf{x}_{n+1})p(\mathbf{z})} \\ = & \alpha\frac{p(\mathbf{z}|\mathbf{x}_{n+1})}{p(\mathbf{z})}p(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n), \end{aligned} \tag{13}$$

where α is a normalization constant. In this case, $\alpha = \frac{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)p(\mathbf{x}_{n+1})}{p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1})}$.

Without loss of generality, let us consider a one-dimensional case for the followings. The solution can be easily extended to a multivariate case since all feature dimensions are assumed to be independent. It can be shown that the posterior distribution in Equation (13) is a Gaussian distribution through induction. Let us assume $p(z|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ is a Gaussian distribution with $\hat{\mu}_n$ and $\hat{\sigma}_n^2$ as mean and variance, respectively. Note that the initial case $p(z|\mathbf{x}_1)$ is guaranteed to be a Gaussian distribution. Let μ_0 and σ_0^2 denote the parameters of the noninformative prior of z. Then, if we take log on both side of Equation (13), we have:

$$\begin{aligned} & \log p(z|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}) \\ = & \log p(z|\mathbf{x}_{n+1}) + \log p(z|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) - \log p(z) + \text{const} \\ = & -\frac{(z - \mu_{n+1})^2}{2\sigma_{n+1}^2} - \frac{(z - \hat{\mu}_n)^2}{2\hat{\sigma}_n^2} + \frac{(z - \mu_0)^2}{2\sigma_0^2} + \text{const} \\ = & -\frac{(z - \hat{\mu}_{n+1})^2}{2\hat{\sigma}_{n+1}^2} + \text{const}. \end{aligned} \tag{14}$$

where "const" refers to the terms irrelevant to z and

$$\hat{\mu}_{n+1} = \hat{\sigma}^2_{n+1}(\frac{\mu_{n+1}}{\sigma^2_{n+1}} + \frac{\hat{\mu}_n}{\hat{\sigma}^2_n} - \frac{\mu_0}{\sigma^2_0}), \tag{15}$$

$$\frac{1}{\hat{\sigma}^2_{n+1}} = \frac{1}{\sigma^2_{n+1}} + \frac{1}{\hat{\sigma}^2_n} - \frac{1}{\sigma^2_0}. \tag{16}$$

Considering $\sigma_0 \to \infty$, we have

$$\hat{\mu}_{n+1} = \frac{\hat{\sigma}^2_n \mu_{n+1} + \sigma^2_{n+1}\hat{\mu}_n}{\sigma^2_{n+1} + \hat{\sigma}^2_n}, \tag{17}$$

$$\hat{\sigma}^2_{n+1} = \frac{\sigma^2_{n+1}\hat{\sigma}^2_n}{\sigma^2_{n+1} + \hat{\sigma}^2_n}. \tag{18}$$

The result means the posterior distribution is a new Gaussian distribution with a smaller variance. Further, we can directly give the solution of fusing n samples:

$$\begin{aligned}
&\log p(z|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \\
&= \log[\alpha p(z|\mathbf{x}_1) \prod_{i=2}^{n} \frac{p(z|\mathbf{x}_i)}{p(z)}] \\
&= (n-1)\log p(z) - \sum_{i=1}^{n} \log p(z|\mathbf{x}_i) + \text{const} \\
&= (n-1)\frac{(z-\mu_0)^2}{2\sigma^2_0} - \sum_{i=1}^{n} \frac{(z-\mu_i)^2}{2\sigma^2_i} + \text{const} \\
&= -\frac{(z-\hat{\mu}_n)^2}{2\hat{\sigma}^2_n} + \text{const}.
\end{aligned} \tag{19}$$

where $\alpha = \frac{\prod_{i=1}^{n} p(\mathbf{x}_i)}{p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)}$ and

$$\hat{\mu}_n = \sum_{i=1}^{n} \frac{\hat{\sigma}^2_n}{\sigma^2_i}\mu_i - (n-1)\frac{\hat{\sigma}^2_n}{\sigma^2_0}\mu_0, \tag{20}$$

$$\frac{1}{\hat{\sigma}^2_n} = \sum_{i=1}^{n} \frac{1}{\sigma^2_i} - (n-1)\frac{1}{\sigma^2_0}. \tag{21}$$

Considering $\sigma_0 \to \infty$, we have

$$\hat{\mu}_n = \sum_{i=1}^{n} \frac{\hat{\sigma}^2_n}{\sigma^2_i}\mu_i, \tag{22}$$

$$\hat{\sigma}^2_n = \frac{1}{\sum_{i=1}^{n} \frac{1}{\sigma^2_i}}. \tag{23}$$

B. Implementation Details

All the models in the paper are implemented using Tensorflow r1.9. Two and Four GeForce GTX 1080 Ti GPUs are used for training base models on CASIA-Webface [10] and MS-Celeb-1M [1], respectively. The uncertainty modules are trained using one GPU.

B.1. Data Preprocessing

All the face images are first passed through MTCNN face detector [7] to detect 5 facial landmarks (two eyes, nose and two mouth corners). Then, similarity transformation is used to normalize the face images based on the five landmarks. After transformation, the images are resized to 112×96. Before passing into networks, each pixel in the RGB image is normalized by subtracting 127.5 and dividing by 128.

B.2. Base Models

The base models for CASIA-Webface [10] are trained for $28,000$ steps using a SGD optimizer with a momentum of 0.9. The learning rate starts at 0.1, and is decreased to 0.01 and 0.001 after $16,000$ and $24,000$ steps, respectively. For the base model trained on Ms-Celeb-1M [1], we train the network for $140,000$ steps using the same optimizer settings. The learning rate starts at 0.1, and is decreased to 0.01 and 0.001 after $80,000$ and $120,000$ steps, respectively. The batch size, feature dimension and weight decay are set to 256, 512 and 0.0005, respectively, for both cases.

B.3. Uncertainty Module

Architecture The uncertainty module for all models are two-layer perceptrons with the same architecture: `FC-BN-ReLU-FC-BN-exp`, where `FC` refers to fully connected layers, `BN` refers to batch normalization layers [2] and `exp` function ensures the outputs σ^2 are all positive values [3]. The first `FC` shares the same input with the bottleneck layer, *i.e.* the output feature map of the last convolutional layer. The output of both `FC` layers are D-dimensional vectors, where D is the dimensionality of the latent space. In addition, we constrain the last `BN` layer to share the same γ and β across all dimensions, which we found to help stabilizing the training.

Training For the models trained on CASIA-WebFace [10], we train the uncertainty module for $3,000$ steps using a SGD optimizer with a momentum of 0.9. The learning rate starts at 0.001, and is decreased to 0.0001 after $2,000$ steps. For the model trained on MS-Celeb-1M[1], we train the uncertainty module for $12,000$ steps. The learning rate starts at 0.001, and is decreased to 0.0001 after $8,000$ steps. The batch size for both cases are 256. For each mini-batch, we randomly select 4 images from 64 different subjects to compose the positive pairs (384 pairs in all). The weight decay is set to 0.0005 in all cases. A Subset of the training data was separated as the validation set for choosing these hyper-parameters during development phase.

Base Model	Representation	LFW	YTF	CFP-FP	IJB-A
Softmax + Center Loss [8]	Original	97.70	92.56	91.13	63.93
	PFE	**97.89**	**93.10**	**91.36**	**64.33**
Triplet [5]	Original	96.98	90.72	**85.69**	**54.47**
	PFE	**97.10**	**91.22**	85.10	51.35
A-Softmax [4]	Original	97.12	**92.38**	89.31	54.48
	PFE	**97.92**	91.78	**89.96**	**58.09**
AM-Softmax [6]	Original	98.32	93.50	90.24	71.28
	PFE	**98.63**	**94.00**	**90.50**	**75.92**

Table 1: Results of CASIA-Net models trained on CASIA-WebFace. "Original" refers to the deterministic embeddings. The better performance among each base model are shown in bold numbers. "PFE" uses mutual likelihood score for matching. IJB-A results are verification rates at FAR=0.1%.

Base Model	Representation	LFW	YTF	CFP-FP	IJB-A
Softmax + Center Loss [8]	Original	97.77	92.34	90.96	60.42
	PFE	**98.28**	**93.24**	**92.29**	**62.41**
Triplet [5]	Original	97.48	92.46	90.01	52.34
	PFE	**98.15**	**93.62**	**90.54**	**56.81**
A-Softmax [4]	Original	98.07	92.72	89.34	63.21
	PFE	**98.47**	**93.44**	**90.54**	**71.96**
AM-Softmax [6]	Original	98.68	93.78	90.59	76.50
	PFE	**98.95**	**94.34**	**91.26**	**80.00**

Table 2: Results of Light-CNN models trained on CASIA-WebFace. "Original" refers to the deterministic embeddings. The better performance among each base model are shown in bold numbers. "PFE" uses mutual likelihood score for matching. IJB-A results are verification rates at FAR=0.1%.

Inference Speed Feature extraction (passing through the whole network) using one GPU takes 1.5ms per image. Note that given the small size of the uncertainty module, it has little impact on the feature extraction time. Matching images using cosine similarity and mutual likelihood score takes 4ns and 15ns , respectively. Both are neglectable in comparison with feature extraction time.

C. Results on Different Architectures

Throughout the main paper, we conducted the experiments using a 64-layer CNN network [4]. Here, we evaluate the proposed method on two different network architectures for face recognition: CASIA-Net [10] and 29-layer Light-CNN [9]. Notice that both networks require different image shapes from our preprocessed ones. Thus we pad our images with zero values and resize them into the target size. Since the main purpose of the experiment is to evaluate the efficacy of the uncertainty module rather than comparing with the original results of these networks, the difference in the preprocessing should not affect a fair comparison. Besides, the original CASIA-Net does not converge for A-Softmax and AM-Softmax, so we add an bottleneck layer to output the embedding representation after the average pooling layer. Then we conduct the experiments by comparing probabilistic embeddings with base deterministic embeddings, similar to Section 5.1 in the main paper. The results are shown in Table 1 and Table 2. Without tuning the architecture of the uncertainty module nor the hyper-parameters, PFE still improve the performance in most cases.

References

[1] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV*, 2016. 2

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[3] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, 2017. 2

[4] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 3

[5] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 3

[6] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 2018. 3

[7] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016. 2

[8] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016. 3

[9] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light cnn for deep face representation with noisy labels. *IEEE Trans. on Information Forensics and Security*, 2015. 3

[10] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv:1411.7923*, 2014. 2, 3