

---

# Cluster and Aggregate: Face Recognition with Large Probe Set Supplementary Material

---

**Minchul Kim**

Department of Computer Science  
Michigan State University  
East Lansing, MI 48824  
kimminc2@msu.edu

**Feng Liu**

Department of Computer Science  
Michigan State University  
East Lansing, MI 48824  
liufeng6@msu.edu

**Anil Jain**

Department of Computer Science  
Michigan State University  
East Lansing, MI 48824  
jain@msu.edu

**Xiaoming Liu**

Department of Computer Science  
Michigan State University  
East Lansing, MI 48824  
liuxm@msu.edu

## A Implementation Details

To train the fusion network  $F$  which is comprised of SIM, CN and AGN, we set the batch size to be 512. We take the pretrained model  $E$ , which is IResNet-101 [2], trained on WebFace4M [15] with ArcFace loss [2] and freeze it without further tuning. For training CAFace, the number of images per identity  $N$  is randomly chosen between 2 and 16 during each step of training, and we take two sets per identity. The intermediate feature for the Style Input Component (SIM) is taken from the block 3 and 4 of the IResNet-101. The number of clusters in CN is varied in the ablation studies and fixed to be 4 for subsequent experiments. The number of layers  $L$  in CN is equal to 2.

We train the whole network end-to-end for 10 epochs with an AdamW optimizer [9]. The learning rate is set to  $1e-3$  and decayed by  $1/10$  at epochs 6 and 9. The weight decay is set to  $5e-4$ . For the loss term, we use  $\lambda_t = 1.0$  and  $\lambda_p = 1.0$  while the efficacy of  $\lambda_p = 1.0$  is ablated with  $\lambda_p = 0.0$  in the ablation studies. For  $\mathbf{f}_{GT}^{(p)}$ , we take the feature embeddings  $\mathbf{f}_i$  extracted from  $E$  for each labeled image in the training data, and average them per identity, with a flip augmentation.

## B Norm Embedding

For an embedding vector  $\mathbf{f}_i$ , the norm is a model dependent quantity, we L2 normalize the feature norm using batch statistics  $\mu_f$  and  $\sigma_f$  and convert it to a bounded integer between  $[-qk, qk]$ .

$$\widehat{\|\mathbf{f}_i\|} = \left\lfloor \left( q * \left( \left\lfloor \frac{\|\mathbf{f}_i\| - \mu_f}{\sigma_f} \right\rfloor_{-k}^k \right) \right) \right\rfloor. \quad (1)$$

Two hyper-parameters,  $q$  and  $k$  control the concentration of the  $\widehat{\|\mathbf{f}_i\|}$  distribution and  $\lfloor \cdot \rfloor_{-k}^k$  refers to clipping the value between  $-k$  and  $k$ .  $\lfloor \cdot \rfloor$  refers to the floor operation to convert the quantity to an integer. Following the convention of Sinusoidal position embedding in [12], we let

$$\mathbf{n}_t(2t) = \sin(\widehat{\|\mathbf{f}_i\|}/10000^{\frac{2t}{c}}), \quad \mathbf{n}_t(2t+1) = \cos(\widehat{\|\mathbf{f}_i\|}/10000^{\frac{2t}{c}}), \quad (2)$$

where  $t$  is the channel index and  $c$  is the dimension of the norm embedding. The resulting  $\mathbf{n}_i \in \mathbb{R}^c$  is a 64-dim vector in our experiments.

## C Additional Performance Results

In this section, we provide additional performance results from IJB-A [6], IJB-B [13], IJB-C [10] and IJB-S [4] dataset with additional backbones.

Table 1: A performance comparison of recent methods on the IJB-A [6] dataset. The  $\pm$  sign refers to the standard deviation calculated from the official 10-fold cross validation splits from the dataset. For recent SoTA backbone models, the performance is saturated above 98.5.

IJB-A [6]	Dataset	Backbone $E$	TAR@FAR=0.001	TAR@FAR=0.01
Naive Average	VGGFace2(3.3M) [1]	ResNet50	89.5 $\pm$ 1.9	95.0 $\pm$ 0.5
QAN [8]	VGGFace2(3.3M) [1]	CNN256	89.3 $\pm$ 3.9	94.2 $\pm$ 1.5
NAN [14]	3M Web Crawl [14]	GoogleNet	88.1 $\pm$ 1.1	94.1 $\pm$ 0.8
RSA [7]	VGGFace2(3.3M) [1]	ResNet50	94.3 $\pm$ 0.8	97.6 $\pm$ 0.6
Naive Average	WebFace4M [15]	IResNet101+ArcFace [2]	98.5 $\pm$ 0.6	99.1 $\pm$ 0.2
PFE [11]	WebFace4M [15]	IResNet101+ArcFace [2]	98.5 $\pm$ 0.6	99.1 $\pm$ 0.2
CFAN [3]	WebFace4M [15]	IResNet101+ArcFace [2]	98.5 $\pm$ 0.5	99.2 $\pm$ 0.2
RSA [7]	WebFace4M [15]	IResNet101+ArcFace [2]	98.6 $\pm$ 0.5	99.1 $\pm$ 0.2
CAFace	WebFace4M [15]	IResNet101+ArcFace [2]	<b>98.7 <math>\pm</math> 0.4</b>	<b>99.2 <math>\pm</math> 0.2</b>

Table 2: A performance comparison of recent methods on the IJB-C [6] dataset. CAFE achieves the best result in IJB-C dataset. We also compare two different backbones ArcFace [2] and AdaFace [5] (CVPR'22). The performance gain is observed in both backbones.

IJB-C [10]	Dataset	Backbone $E$	TAR@FAR=1e-3	TAR@FAR=1e-4	TAR@FAR=1e-5
Naive Average	WebFace4M [15]	IResNet101+ArcFace [2]	97.30	95.78	92.60
PFE [11]	WebFace4M [15]	IResNet101+ArcFace [2]	97.53	96.33	94.16
CFAN [3]	WebFace4M [15]	IResNet101+ArcFace [2]	97.55	96.45	94.40
RSA [7]	WebFace4M [15]	IResNet101+ArcFace [2]	97.49	96.49	94.58
CAFace	WebFace4M [15]	IResNet101+ArcFace [2]	<b>97.99</b>	<b>97.15</b>	<b>95.78</b>
Naive Average	WebFace4M [15]	IResNet101+AdaFace [5]	97.63	96.42	94.47
CAFace	WebFace4M [15]	IResNet101+AdaFace [5]	<b>98.08</b>	<b>97.30</b>	<b>95.96</b>

Table 3: An additional performance on the IJB-B [6] dataset. We compare two different backbones ArcFace [2] and AdaFace [5] (CVPR'22).

IJB-B [13]	Dataset	Backbone $E$	TAR@FAR=1e-3	TAR@FAR=1e-4	TAR@FAR=1e-5
Naive Average	WebFace4M [15]	IResNet101+ArcFace [2]	96.1	94.30	89.53
CAFace	WebFace4M [15]	IResNet101+ArcFace [2]	<b>96.91</b>	<b>95.53</b>	<b>92.29</b>
Naive Average	WebFace4M [15]	IResNet101+AdaFace [5]	96.66	94.84	90.86
CAFace	WebFace4M [15]	IResNet101+AdaFace [5]	<b>96.97</b>	<b>95.78</b>	<b>92.78</b>

Table 4: An additional performance result on IJB-S [4] dataset with two different backbones, ArcFace [2] and AdaFace [5] (CVPR'22). AdaFace [5] combined with our proposed CAFE achieves a large margin improvement in IJB-S.

Method	$E$	Surveillance-to-Single			Surveillance-to-Booking			Surveillance-to-Surveillance		
		Rank-1	Rank-5	1%	Rank-1	Rank-5	1%	Rank-1	Rank-5	1%
Naive Average	ArcFace	69.26	74.31	57.06	70.32	75.16	56.89	32.13	46.67	5.32
CAFace	ArcFace	<b>71.61</b>	<b>76.43</b>	<b>62.21</b>	<b>72.72</b>	<b>77.41</b>	<b>62.68</b>	<b>36.51</b>	<b>49.59</b>	<b>8.78</b>
Naive Average	AdaFace	70.42	75.29	58.27	70.93	76.11	58.02	35.05	48.22	4.96
CAFace	AdaFace	<b>72.91</b>	<b>77.14</b>	<b>62.96</b>	<b>73.39</b>	<b>78.04</b>	<b>63.61</b>	<b>39.25</b>	<b>50.47</b>	<b>7.65</b>

The size of the probes  $N$  in each dataset increases in the order of IJBA [6], IJBB [13], IJB-C [10] and IJB S [4]. As the probe size increases, the role of a feature fusion model also increases. As noted in Fig.1 c) of the main paper, previous methods either fail to model the intra-set relationship or scale to large  $N$ , which results in a suboptimal performance with an increasing probe size. The plot of the relative performance increase over the naive average baseline shows that for CAFE, as the set size increases, the performance gain also increases. The relative performance gain for Fig.1 c) is calculated as  $\frac{Method - Naive}{Naive} * 100\%$  where the metrics for each dataset are TAR@FAR=0.001 for IJB-A, TAR@FAR=1e-4 for IJB-B and IJB-C, and the average of 9 metrics across all 3 protocols for IJB-S.

## D Resource and Efficiency Comparison

We report the FPS (frames per second) to give the estimation of how much resource the feature fusion framework takes with respect to the backbone  $E$ . For the table below, we use the backbone of IResNet-101 [2]. We measured the FPS with Nvidia RTX3090. It is equipped with a GPU memory of 24 GB. For measuring the time, we feed the random array as an input to the model and simulate the run for 1,000 times. In Tab. 5, we first show the FPS for the backbone  $E$ . The FPS increases with batch-size due to the efficiency of GPU architecture. We take 1,288 FPS as the FPS for the backbone and measure the relative FPS of the fusion models  $F$  with respect to the backbone, *i.e.*  $\frac{FPS(F)}{FPS(E)}$ .

In Tab. 6, we show  $\frac{FPS(F)}{FPS(E)}$  of various feature fusion models with the varied set size  $N$ . First, note that the feature fusion model’s inference speed is always faster than the backbone model, *i.e.*  $\frac{FPS(F)}{FPS(E)} > 1$ . In practice, we would like the fusion time to be a fraction of the backbone inference time. Secondly, we show the maximum set size  $N$  each method can take. Note that methods without intra-set relationships, PFE [11] and CFAN [3], are computationally very fast and require little memory. Therefore, it can take many samples together (large  $N$ ) during inference. In contrast, the maximum set size  $N$  for RSA [7] is 384 because the intra-set attention with the feature map is a memory-intensive module. CAFace is fast and uses relatively little memory, allowing the maximum set number to be  $N = 12,000$ .

Note the ability to perform sequential inference is different from large  $N$ . For instance, with CAFace, we can split a set of size 64,000 with a batch size of 64 and run 1,000 sequential inferences, without sacrificing the performance. It is evident in the high performance of IJB-S dataset, where we adopt the batch size of 256.

Table 5: FPS for the face recognition backbone model IResNet-101. Higher the FPS, the faster the inference speed per image.

	Batch Size	FPS
Backbone (Batchsize: 1)	1	91
Backbone (Batchsize: 256)	256	<b>1,288</b>

Table 6: A table of relative FPS of the fusion model w.r.t. the FPS of the backbone. We compare various fusion models with varied input size  $N$ . As  $N$  increases, it requires more GPU memory as well. Max  $N$  refers to the maximum number of images that can be in a set without causing the out of memory error (OOM). The higher the  $\frac{FPS(F)}{FPS(E)}$ , the faster the fusion method.

$\frac{FPS(F)}{FPS(E)}$	Max $N$	$N = 16$	$N = 32$	$N = 64$	$N = 256$	$N = 512$
PFE	115,200	21.8x	44.1x	86.3x	360.1x	2133.6x
CFAN	115,200	82.6x	158.7x	268.8x	544.1x	664.2x
<b>CAFace</b>	<b>12,000</b>	<b>4.2x</b>	<b>8.2x</b>	<b>16.4x</b>	<b>64.4x</b>	<b>129.3x</b>
RSA	384	6.9x	13.1x	9.2x	3.1x	<b>OOM</b>

## E Training Progress and Learned Assignment

To see how the assignment behavior changes during training, we plot the entropy of the assignment map  $\mathbf{A} \in \mathbb{R}^{M \times N}$  over the training epochs. We note that each  $j$ -th cluster is a weighted average of individual  $N$  samples. Therefore, if all samples are contributing equally to the  $j$ -th cluster, then the entropy of  $\mathbf{A}$  for each row would be high. When a few samples' contribution is larger than the others (*i.e.*,  $\mathbf{A}$  is sparse) then the entropy would be low. We use entropy as a proxy of how sparse is the influence of samples for each cluster.

The entropy is calculated as

$$\sum_{j=1}^M \sum_{i=1}^N -p_{j,i} \log(p_{j,i}),$$

where  $p_{j,i} = \mathbf{A}_{j,i} / \sum_{i=1}^N \mathbf{A}_{j,i}$ . In other words, it is the mean of the row-wise entropy of the normalized assignment map. Lower entropy value tells you that the cluster features are deviating from a simple average of all samples. In Fig. 1, we show the plot of the mean entropy over the training progression using the IJB-B dataset [13]. The value decreases steeply during the first few epochs, indicating that the clustering mechanism is quickly deviating from the simple averaging of the given samples.

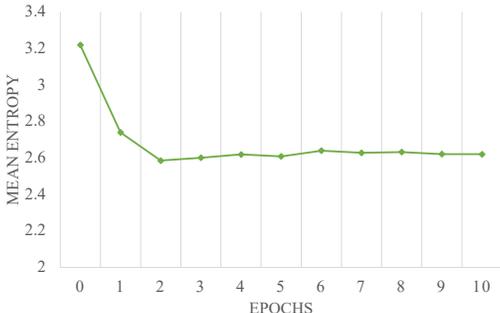


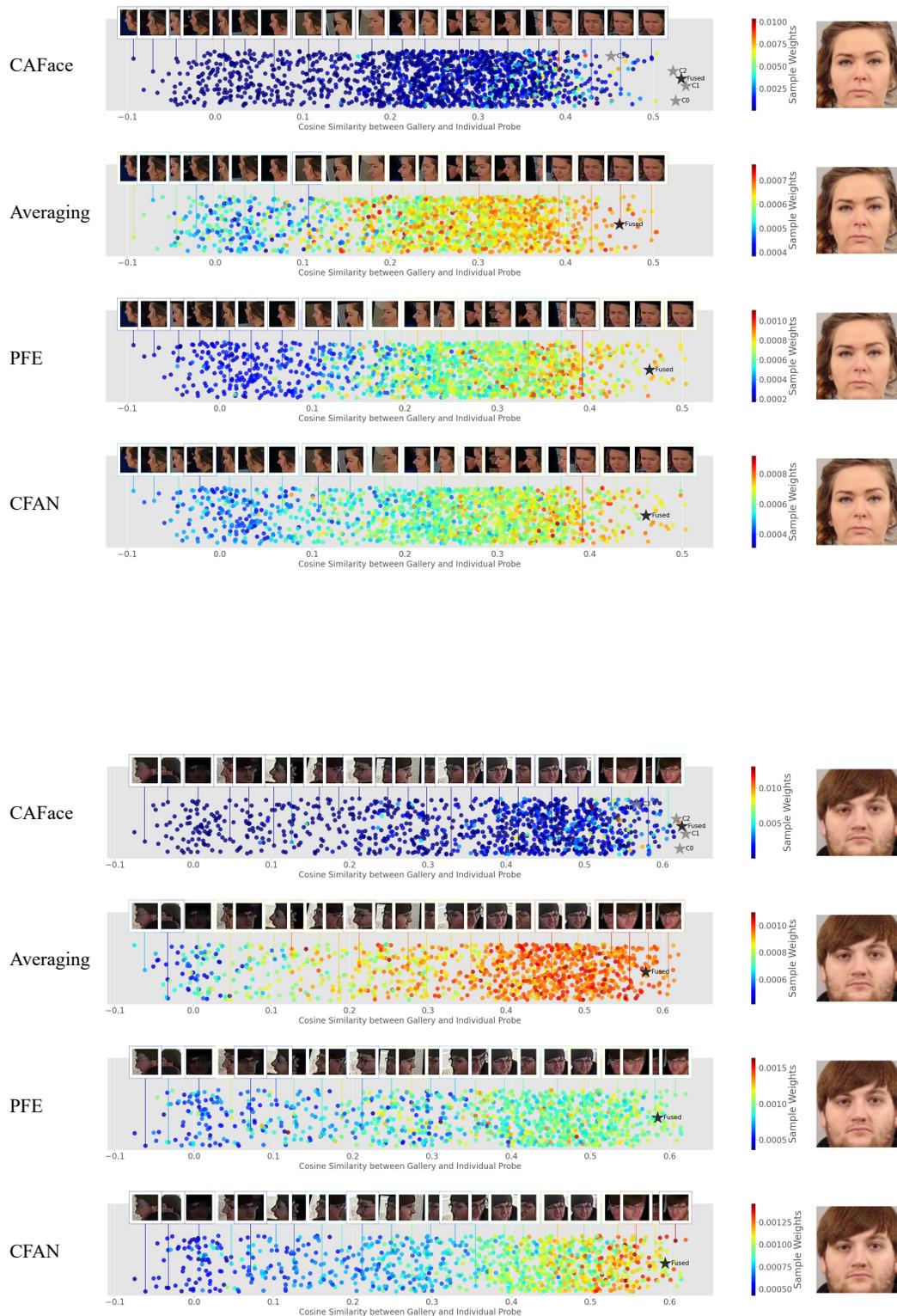
Figure 1: A plot of mean entropy during training. The samples used are random 200 probes taken from the IJB-B [13] dataset.

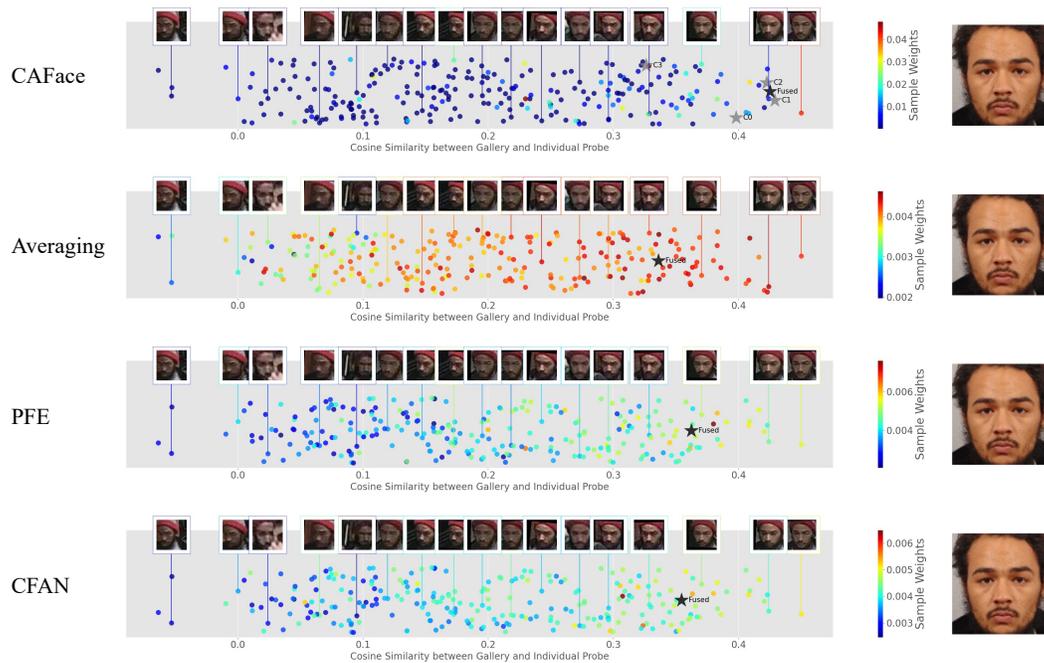
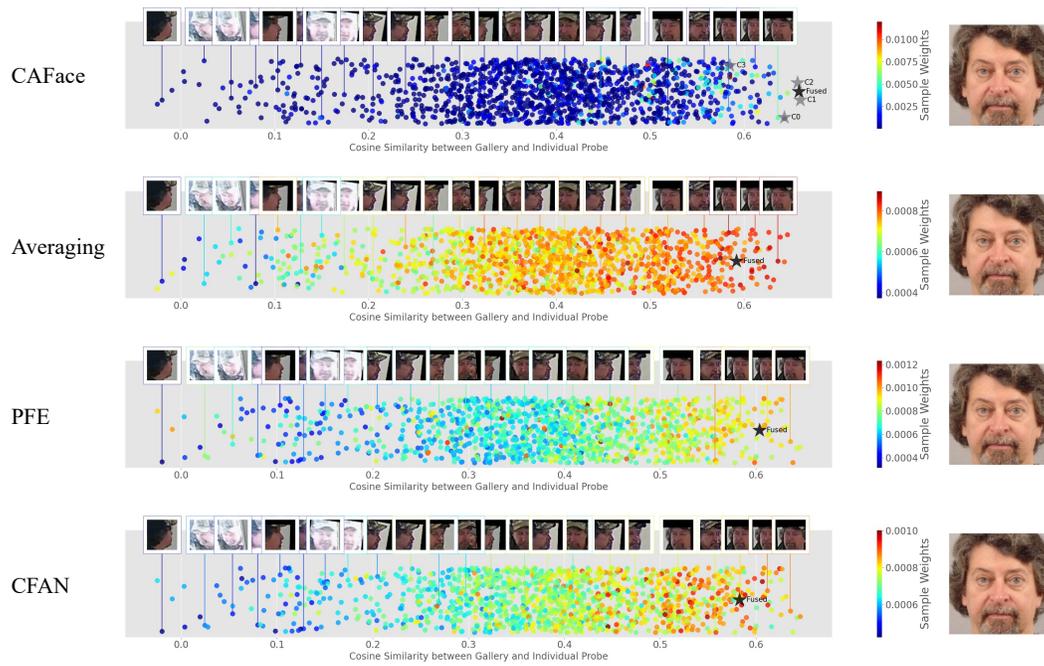
## F Weight Visualization

We show a few examples of the weight visualizations of different methods. The weights for CAFace are calculated as

$$w_i = \frac{\sum_j^M \mathbf{A}_{j,i} \sum_{c=1}^C (\mathbf{P}_{j,c}/C)}{z},$$

the sum of the contributions each sample makes to each cluster, weighed by the importance of the cluster.  $C$  is the dimension of  $\mathbf{f}$ , which is 512 in our backbone.  $M$  is the number of clusters.  $z$  is the normalization constant to make the  $\sum_{i=1}^N w_i = 1$ . For the Averaging, the weights are the normalized feature norms. For PFE and CFAN, the weights are the output of the respective modules. Note that RSA does not have a weight estimation, as it directly estimates the fused output as opposed to estimating the weights. The circles in the plot represent individual probe images in IJB-S and the color represents the magnitude of the weights. The horizontal axis represents the similarity of individual probe images to the gallery shown on the right. The vertical axis exists only to scatter the points. Note that for both PFE and CFAN, the weight estimation is based on a single image.







## References

- [1] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *FG*, 2018.
- [2] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. In *CVPR*, 2019.
- [3] Sixue Gong, Yichu Shi, Nathan D Kalka, and Anil K Jain. Video face recognition: Component-wise feature aggregation network (C-Fan). In *ICB*, 2019.
- [4] Nathan D Kalka, Brianna Maze, James A Duncan, Kevin O’Connor, Stephen Elliott, Kaleb Hebert, Julia Bryan, and Anil K Jain. IJB-S: IARPA Janus Surveillance Video Benchmark. In *BTAS*, 2018.
- [5] Minchul Kim, Anil K Jain, and Xiaoming Liu. AdaFace: Quality adaptive margin for face recognition. In *CVPR*, 2022.
- [6] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, and Anil K Jain. Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark-A. In *CVPR*, 2015.
- [7] Xiaofeng Liu, Zhenhua Guo, Site Li, Lingsheng Kong, Ping Jia, Jane You, and BVK Kumar. Permutation-invariant feature restructuring for correlation-aware image set-based recognition. In *ICCV*, 2019.
- [8] Yu Liu, Junjie Yan, and Wanli Ouyang. Quality aware network for set to set recognition. In *CVPR*, 2017.
- [9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [10] Brianna Maze, Jocelyn Adams, James A Duncan, Nathan Kalka, Tim Miller, Charles Otto, Anil K Jain, W Tyler Niggel, Janet Anderson, Jordan Cheney, and Patrick Grother. IARPA Janus Benchmark-C: Face dataset and protocol. In *ICB*, 2018.
- [11] Yichun Shi and Anil K Jain. Probabilistic face embeddings. In *ICCV*, 2019.
- [12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [13] Cameron Whitelam, Emma Taborsky, Austin Blanton, Brianna Maze, Jocelyn Adams, Tim Miller, Nathan Kalka, Anil K Jain, James A Duncan, Kristen Allen, et al. IARPA Janus Benchmark-B face dataset. In *CVPRW*, 2017.
- [14] Jiaolong Yang, Peiran Ren, Dongqing Zhang, Dong Chen, Fang Wen, Hongdong Li, and Gang Hua. Neural aggregation network for video face recognition. In *CVPR*, 2017.
- [15] Zheng Zhu, Guan Huang, Jiankang Deng, Yun Ye, Junjie Huang, Xinze Chen, Jiagang Zhu, Tian Yang, Jiwen Lu, Dalong Du, et al. WebFace260M: A benchmark unveiling the power of million-scale deep face recognition. In *CVPR*, 2021.