Towards Flops-constrained Face Recognition

Yu Liu* Guanglu Song* Manyuan Zhang* Jihao Liu*
Yucong Zhou Junjie Yan
yuliu@ee.cuhk.edu.hk
{songguanglu, zhangmanyuan, liujihao}@sensetime.com
The Chinese University of Hong Kong
SenseTime Research

Abstract

Large scale face recognition is challenging especially when the computational budget is limited. Given a flops upper bound, the key is to find the optimal neural network architecture and optimization method. In this article, we introduce the solutions of team 'trojans' for the ICCV19 - Lightweight Face Recognition Challenge [2]. Our team mainly focuses on the two 'large' tracks, image-based and video-based, respectively. The submissions of these two tracks are required to be one single model with computational budget no higher than 30 GFlops. We introduce a network architecture 'Efficient PolyFace', a novel loss function 'ArcNegFace', a novel frame aggregation method 'QAN++', together with a bag of useful tricks in our implementation (augmentations, regular face, label smoothing, anchor finetuning, etc.). Our basic model, 'Efficient PolyFace', takes 28.25 Gflops for the 'deepglint-large' image-based track, and the 'PolyFace+QAN++' solution takes 24.12 Gflops for the 'iQiyi-large' video-based track. These two solutions achieve 94.198% @ 1e-8 and 72.981% @ 1e-4 in the two tracks respectively, which are the state-of-the-art results[1] in this competition.

1. Lightweight Face Recognition Challenge

The ICCV19-Lightweight Face Recognition Challenge [2] is one of the most strict competitions in open-set face recognition. It requires the strict consistency of training data [4], face detector [3] and alignment method between different submissions. There are four tracks in this competition: small image-based, large image-based, small video-based and large video-based. The computational budged is 1Gflops and 30Gflops for the small and large tracks respectively.

2. Image-based baseline model

We adopt two different CNN architectures R100 [1] and a proposed PolyFace as our base models. The input sizes of the two basic architectures are both 112×112 as required by the challenge [2].

PolyFace. Similar to the structure of PolyNet [11], the basic PolyFace is designed by repeating its basic blocks. Details of the basic blocks are shown in Fig 1. In the stem block of the proposed PolyFace, the spatial size is first upsampled to 235×235 and then downsized to 112×112 by an upsampling and a convolutional layer, which we call 'stem-enrichment block'. The data flow in the whole PolyFace is:

```
Stem block -- A × blockA -- blockA2B
-- B× blockB -- blockB2C -- C× blockC.
```

At the end of all backbones, a fully connected layer with 256 out-channels is adopted to generate the representation, followed by a `BatchNorm1d` layer. The block number of [A,B,C] in base model is [10,20,10].

Training details. During the training process of the base models, 16 GPUs are used to enable a global batch size of 1,024. Synchronized BN is used with group size 1. The total training iterations is set to 100,000, and the initial learning rate is 0.001 and warms up to 0.4 during the first 10,000 iterations. The weight decay is set to 1e-5 and momentum is set to 0.9. Dropout with drop rate of 0.4 for the final embedding is used to prevent overfitting.

The results of two base models on the challenge test server [2] are shown in Tab 1.

*They contributed equally to this work

[1]The 72.981% result achieves the 1st place on the IQIYI-large track and the 94.198% achieves the 2nd place on deepglint-large. However, the result on deepglint-large needs further deliberation. Note that our 94.189% result on deepglint-large is adjusted by AdaBN, which uses image-level information of test set. For a fair comparison, the accuracy of Efficient PolyFace w/o AdaBN is 93.801% as shown in Tab. 4

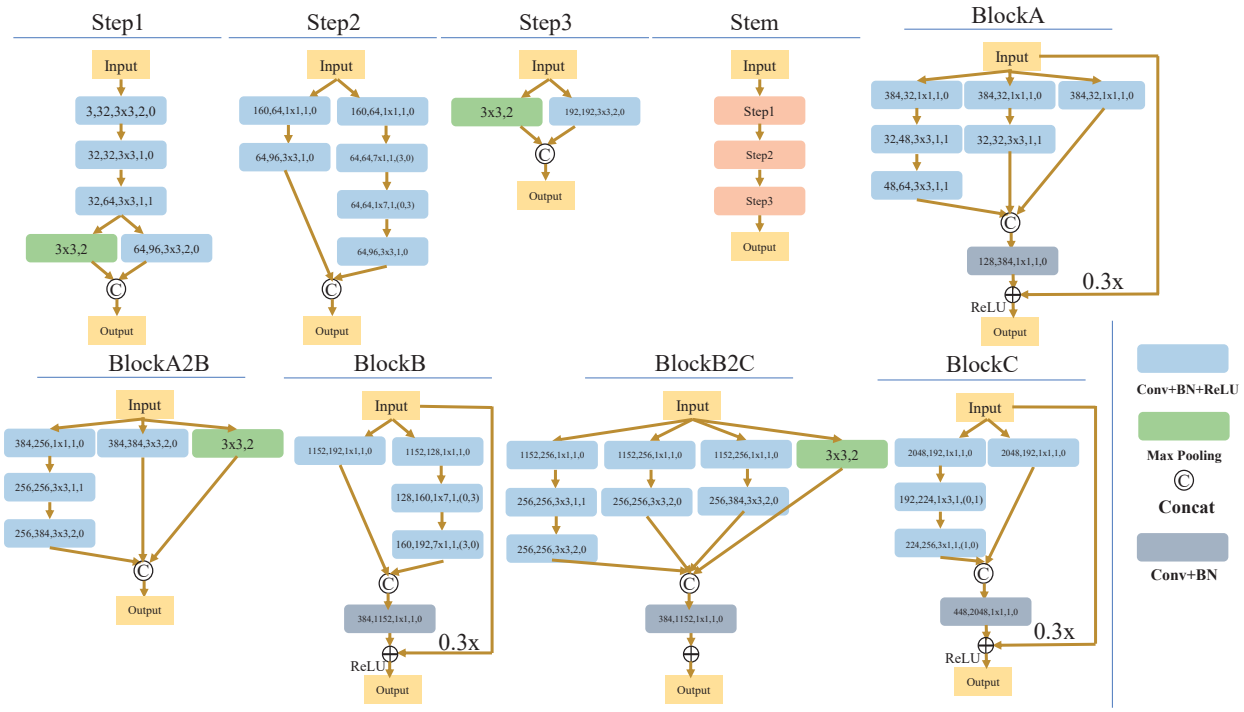

Figure 1. The details of blocks in PolyFace. The numbers in block $Conv+BN+ReLU$ represent the input channel, output channel, kernel size, stride, and padding. The numbers in block $Max\ Pooling$ represent the kernel size and stride. The numbers in block $Conv+BN$ represent the input channel, output channel, kernel size, stride, and padding.

Model	Flops	Loss	TPR@FPR=1e-8
R100	24.22G	ArcFace	90.972
PolyNet	16.62G	ArcFace	90.829

Table 1. The comparison between different base models. The Flops is computed by the public tool in https://github.com/Swall0w/torchstat (the total MAdd in the public tool).

3. New loss function: ArcNegFace

We introduce a new robust loss named ArcNegFace in this section. Unlike most of the recent novel losses that try to find an 'optimal' logits curve to regularize the margin between embedding and class anchors, ArcNegFace takes the distance between anchors into consideration.

Define θ_{y_i} as the angle between the feature f with label y_i and the anchor weight W_{y_i}, the original ArcFace can be defined as:

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

where hyperparam s and m represent the scale and margin. In order to utilize hard negative mining and weaken the influence of the error labeling, we improve the ArcFace to ArcNegFace formulated as:

$$L = -\frac{1}{n}\sum_{i=1}^{N} log\frac{e^{s(cos(\theta_{y_i}+m))}}{e^{s(cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n} e^{s(t_{j,y_i}cos\theta_j+t_{j,y_i}-1)}} \quad (2)$$

where t_{j,y_i} is $\mathbf{G(C_j, C_{yi})}$, C_j and C_{yi} mean the cosθ_j and cos(θ_{y_i}+m). The function $G(\cdot,\cdot)$ is the Gaussian function which is formulated as:

$$G(x,y) = \alpha * e^{-\frac{(x-y-\mu)^2}{2\sigma}} \quad (3)$$

where α, μ and σ are set to 1.2, 0 and 1, respectively. The performance of ArcNegFace is shown in Tab 2

Model	Loss	TPR@FPR=1e-8
PolyNet	ArcFace	90.829
PolyNet	ArcNegFace	91.639

Table 2. The comparison between different loss functions.

4. Efficient PolyFace

Inspired by the idea of efficientnet [10], we launch a NAS processing to expand the basic models in depth and

Block number	Channel number	TPR@FPR=1e-8
[3,13,30,3]	[64,128,256,512]	88.652
[3,13,30,3]	[72,144,288,576]	90.243
[3,16,37,3]	[65,130,260,520]	90.188
[3,20,46,3]	[59,118,236,472]	89.954
[3,25,57,3]	[53,106,212,424]	89.875
[3,13,50,3]	[61,122,244,488]	89.789
[3,9,19,3]	[84,168,336,672]	89.734
[3,9,31,3]	[74,148,296,592]	89.699

Table 3. The performance of different modified R100 models.

width with the constraint of the computation budget. Some selected results on R100 are shown in Tab 3. Note that all of the experiments are trained under the same basic setting. Finally, we found one of the expanded PolyFace models outperforms all searched candidates with the same Flops (∼28 Gflops), so we adopt it, called Efficient PolyFace, as the final backbone [2]. Some selected results are shown in Tab 7.

Model	AdaBN	TPR@FPR=1e-8
Efficient PolyFace		93.801
Efficient PolyFace ABN	$\surd$	**94.198**

Table 4. Performance of AdaBN. The performance 94.198 is the final submission on the leaderboard.

Model	margin	TPR@FPR=1e-8
PolyNet	0.5	90.829
PolyNet	0.3	91.332

Table 5. The performance of different margin based on ArcFace.

5. Bag of tricks

5.1. Anchor finetuning

We introduce a new regularization term named $anchor\ finetuning$. Given a convergent model, we extract the features of the training set and re-init the weight W in the classification layer by the mean feature of the corresponding identity. Then, the model will be finetuned based on this as shown in Tab 6.

5.2. Scale & Shift augmentations

Data augmentation is used during the training process for all settings. The original image will be re-scaled and shifted within $\pm 1\%$ randomly. The performance is shown in Tab 6.

5.3. Color jitter

The brightness, contrast, and saturation are set to 0.125 when adding color jitter.

[2]Model architecture and parameters will be open-source

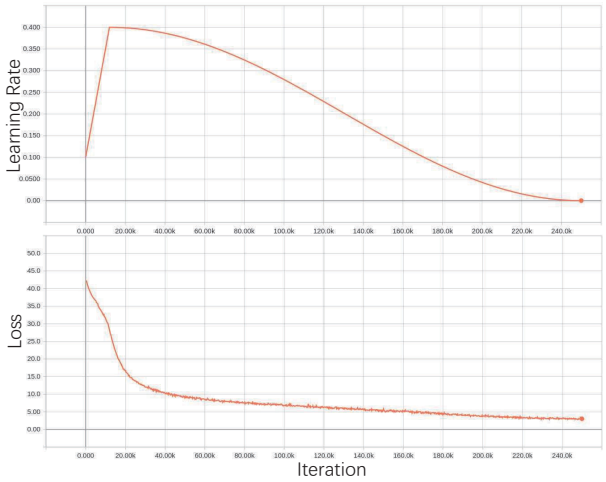

Figure 2. The details of cosine decay.

5.4. Flip strategy

The flip strategy is adopted during the training stage. During the inference stage, we extract the features for both the original and the flipped image. The final feature is the average of them. Results are shown in Tab 6.

5.5. Regular face

Regular face [12] is adapted to constrain the inter-class distance, but we find it can rarely bring improvement while consuming a large memory.

5.6. Label smooth

We explore the label smooth strategy, which is widely used in ImageNet classification. The result is shown in Tab 6.

5.7. AdaBN

Considering the domain shift between the training set and the testset, we perform the AdaBN [7] on the convergent model to improve its performance. Results are shown in Tab 4.

5.8. Modification of margin

We modify the margin in ArcFace and it brings a few improvements as shown in Tab 5.

5.9. Cosine learning rate and stochastic depth

We explore the cosine learning rate decay and stochastic depth [6] to achieve further gain. The keep rate in stochastic depth is set to 0.8 in all experiments. The function of learning rate *w.r.t.* iteration is shown in Fig 2, and results are shown in Tab 7. The losses during the training of basic PolyFace is shown in Fig 2.

Model	ArcNegFace	Scale&Shift aug	Flip	Regular Face [12]	Label smooth	Fc finetune	Arch finetune [5]	TPR@FPR=1e-8
R100	√							81.503
R100	√	√						80.59
R100	√					√		81.628
R100	√	√		√				80.819
R100	√	√			√			81.085
R100	√	√			√		√	81.272
R100	√	√	√		√			81.922
R100	√		√		√		√	81.638

Table 6. The comparison of the different training strategy. Note that the performance is evaluated on the deepglint-large without cleaning up error label.

Model	Flops	Blocks	Cosine decay	Stochastic depth	Color jitter	TPR@FPR=1e-8
PolyNet [11]	16.62G	[10,20,10]	√	√	√	93.066
PolyFace	24.04G	[20,30,20]	√	√	√	93.729
Efficient PolyFace	28.25G	[23,38,23]	√	√	√	93.801

Table 7. The performance of cosine decay and stochastic depth based on ArcNegFace. The $Scale\&Shift\ aug$ and $Flip$ are adopted in these experiments. ArcNegFace with margin 0.3 is used.

6. Enhanced quality aware network for video face recognition

To generate the robust video representation for set-to-set recognition in IQIYI track [2], inspired by QAN and RQEN [8, 9], we propose a new quality estimation strategy called enhanced quality aware network (QAN++) to approximate the quality of each image. The representation of the image set can be aggregated by the weighted sum of frame representations with the assistant of the image quality.

Different from the subjective quality judgment of image, our method assigns the image quality from the characteristics of feature discrimination. Define the dataset D with C identities and the weight anchor $W_i, i \in [1, C]$ in the final classification layer, the quality of image I with ID c can be computed by:

$$Q_I = \frac{cos(F_I, W_c)}{max\{cos(F_I, W_j)|j \in [1, C], j \neq c\}} \tag{4}$$

The image quality is computed on the training set and in order to obtain the image quality during the inference stage, we add a lightweight quality generation branch to regress the quality value computed on the training set. To better regress the quality, we normalize it as:

$$Q_I = \sigma(\frac{Q_I - mean(Q)}{std(Q)}) \tag{5}$$

where $\sigma(\cdot)$, $mean(Q)$ and $std(Q)$ mean the sigmoid function, mean value and standard deviation value in the whole training set respectively. The L2 loss is adopted as the training loss.

During the inference stage, given the video $I_i, i \in [1, n]$ where n means the total image number and the corresponding feature representation F_i, we extract the quality value Q_i of I_i. The quality value will be re-scaled by:

$$Q_i = K \cdot Q_i + B \tag{6}$$

$$K = \frac{1}{max\{Q_i\} - min\{Q_i\}}, i \in [1, n] \tag{7}$$

$$B = 1 - K \cdot max\{Q_i\}, i \in [1, n] \tag{8}$$

Finally, the video-level feature can be aggregated by:

$$F = \sum_{i}^{n} \frac{Q_i \cdot F_i}{Q_i}, i \in [1, n] \tag{9}$$

If the image number n in the image set is less than 3, we directly adopt Eq 9 to aggregate them without re-scaling the quality value.

6.1. Performance of different aggregation strategies

We evaluate the effectiveness of the proposed quality estimation strategy on IQIYI in LFR. Results are shown in Tab 8. We embed a new quality branch into PolyFace. The new branch looks like a tiny version of ResNet-18. The block number in each stage is $[2, 2, 2, 2]$ and the channel number in each stage is set to [8,16,32,48]. We add a fully connected layer with output number 1 after the global average pooling to regress the quality. The flops of the quality net is 81.9 Mflops and the input is the same as the PolyFace.

Model (w/o ABN)	Deepglint	aggregation	IQIYI
R100	92.433	Avg	65.843
R100	92.433	Weighted Sum	67.381
R100	92.433	Top1 Quality	65.217
R100	92.433	QAN++	69.048
PolyFace	93.729	QAN++	**72.981**

Table 8. Comparison with different quality strategies on IQIYI-large track in LFR. The performance 72.981 is the final submission on the leaderboard.

7. Conclusion

In this article, we show the details of our solution to ICCV19-LRF challenge. For the image-based and video-based tracks, We introduce a new backbone Efficient PolyFace and a new loss function ArcNegFace. For the video based track, we propose a novel quality estimator QAN++ to generate quality score for each frame. Besides, we also explore some useful tricks in face recognition model. Results on the challenge test server demonstrate the effectiveness of the proposed methods.

References

[1] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[2] J. Deng, J. Guo, D. Zhang, Y. Deng, X. Lu, S. Shi, and S. Zafeiriou. Lightweight face recognition challenge. In *In Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1, 4

[3] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. *arXiv preprint arXiv:1905.00641*, 2019. 1

[4] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*, pages 87–102. Springer, 2016. 1

[5] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 4

[6] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 3

[7] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016. 3

[8] Y. Liu, J. Yan, and W. Ouyang. Quality aware network for set to set recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 4

[9] G. Song, B. Leng, Y. Liu, C. Hetang, and S. Cai. Region-based quality estimation network for large-scale person re-identification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 4

[10] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2

[11] X. Zhang, Z. Li, C. Change Loy, and D. Lin. Polynet: A pursuit of structural diversity in very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 718–726, 2017. 1, 4

[12] K. Zhao, J. Xu, and M.-M. Cheng. Regularface: Deep face recognition via exclusive regularization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3, 4