Fact sheet: ECCV 2020 ChaLearn Looking at People 1st Fair Face Recognition Challenge

This is the fact sheet's template for the ECCV 2020 ChaLearn Fair Face Recognition Challenge [1]. Please fill out the following sections carefully in a scientific writing style. Then, send the compressed project (in `.zip` format), i.e., the generated PDF, `.tex`, `.bib` and any additional files to `juliojj@gmail.com`, and put in the Subject of the email "ECCVW 2020 FairFaceRec Challenge / Fact Sheets", following the schedule and instructions provided in the Challenge webpage [1] (post-challenge/fact sheets).

I. Team details

- Team leader name: Shengyao Zhou
- Username on Codalab: paranoidai
- Team leader affiliation: RuiYan Technology
- Team leader email: 197479645@qq.com
- Name of other team members (and affiliation): Junfan Luo - RuiYan Technology, Junkun Zhou - RuiYan Technology and Xiang Ji - RuiYan Technology
- Team website URL (if any): `http://www.ruiyanai.com/`

II. Contribution details

A. Title of the contribution

Fairface recognition is a challenging task due to high variances between diffrent attributes and unbalancement of data. In this work, we provide an approach to make a fairface recognition by using asymmetric-arc-loss training and multi-step finetuning. First,we train a general model, and then, we make a mutli-step finetuning to get higher auc and lower bias. Besides, we propose another viewpoint on reducing the bias and bag of tricks such as reranking, boundary cut and hard-sample model fusion to improve the performance.

B. Introduction and Motivation

Face recogniton has been widely used and researched and a lot of class-level losses such as Softmax, SphereFace[2], CosineFace [3] and ArcFace [4] are used to improve the perfomance. All of these losses are trying to minimize between-class similarity s_n and maximize within-class similarity s_p. However, we don't always need to minimize between-class similarity extremely since there are also similar faces between class, in this situation, try to minimize the between-class similarity extremely may lead to noise in gradient and potentially lead to worse convergence. Besides, the previous face-recognition approach focus more on the auc on the whole test-set and less on bias between attributes. Based on these ideas, wo proposed a fairface recognition approach aiming at higher accuracy and lower bias. Our major contribution can be summarized into four aspects:

- **First, an asymmetric-arc-loss.** From the previous class-level loss analysis, we propose an asymmetric-arc-loss which is a combination of arc-face loss and circle-loss. Besides, we modify the loss contribution from negtive pairs similarities and make it asymmetric to positive similarities, which means we don't minimize the negtive similirity extremely, which finally decrease the gradient contibution from easy negtive samples.

- **Second, a multi-step finetuning.** We propose a multi-step finetuing method to minimize the bias between different protected attributes and this is controllable and stable in improving the model's performance on most discriminated protected-attribute data.

- **Third, bag of tricks.** We use bag of tricks such as reranking, boundary cut and hard-sample model fusion to get higher accuracy and lower bias. And the hard-sample model fusion are quite significant for bias mitigation.

- **Finally, another viewpoint on bias mitigation.** We give another viewpoint on bias mitigation. And it's easy to impement and can decrease the bias even as whatever you want.

C. Representative image / workflow diagram of the method

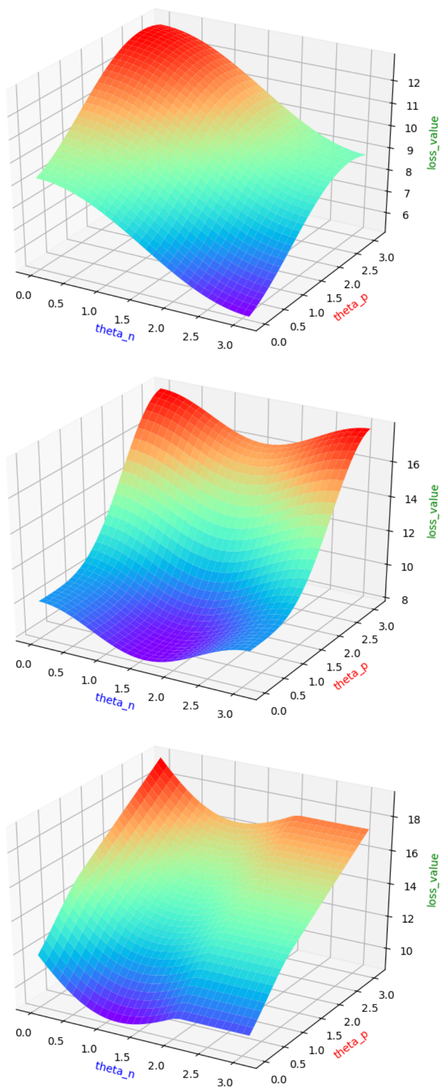

Fig. 1. Loss value with θ_n and θ_p for arc-face-loss, circle-loss and asymmetric-arc-loss.

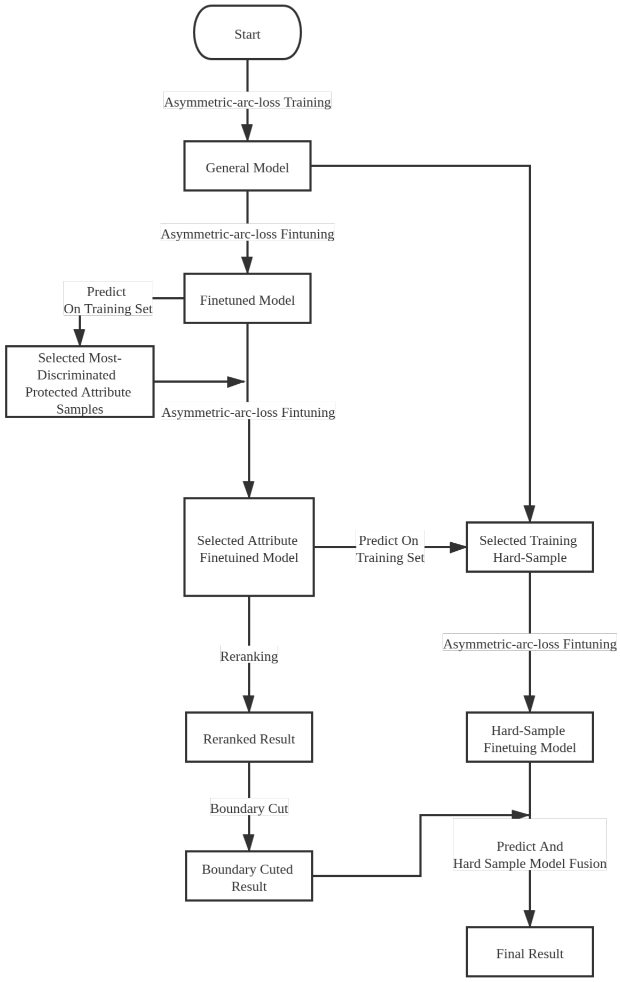

Fig. 2. Pipeline for the whole process.

TABLE I

BACKBONE AND RESULTS USING ASYMMETRIC-ARC-LOSS ON VALIDATION.

Backbone	PosiBias	NegBias	Auc
Mobilefacenet	0.012916	0.017778	0.982363
Resnet50	0.009215	0.010823	0.988492
Resnet101	0.005070	0.006807	0.992260
ResNeSt101	0.005939	0.007511	0.990803

TABLE II

LOSS AND RESULTS USING RESNET101 ON VALIDATION

Loss	PosiBias	NegBias	Auc
Arcface	0.021367	0.017595	0.976372
Circle-loss	0.008559	0.009488	0.991284
Asymmetric-arc-loss	0.005070	0.006807	0.992260

TABLE III

FINETUNING STEP AND POSTPROCESS ON VALIDATION

Step	PosiBias	NegBias	Auc
Asymmetric-arc-loss Training	0.005070	0.006807	0.992260
Asymmetric-arc-loss Finetuing	0.004988	0.005699	0.994518
Select Attribute Finetuing	0.005414	0.001250	0.995319
Select Attribute Finetuing with Reranking and BoundaryCut	0.002707	0.000697	0.996075

TABLE IV

FINETUNING STEP AND POSTPROCESS ON TEST

Step	PosiBias	NegBias	Auc
Asymmetric-arc-loss Finetuing with Reranking and BoundaryCut	0.000299	0.000115	0.999899
Select Attribute Finetuing with Reranking and BoundaryCut	0.000273	0.000079	0.999910
Select Attribute Finetuing with Reranking and BoundaryCut and Hard-Sample Fusion	0.000012	0.000059	0.999966

D. Detailed method description

Just as the Fig. 2 shows, our solution can be summarized to these 5 steps:

1) Step1.Train a general model: In this step, we proposed asymmetric-arc-loss for training. Let's start with the arcface loss [4]:

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

subject to

$$W_j = \frac{W_j}{\|W_j\|}, x_i = \frac{x_i}{\|x_i\|}, \cos\theta_j = W_j^T x_i$$

We assume θ_{y_i} as θ_p and others as θ_n.It's easy to analyze that the loss is Monotonically increasing to the θ_p while $\theta_p + m < \pi$ and Monotonically decreasing to θ_n, so, as shown in Fig.1, it's convergence target is to maximize θ_n and to minimize θ_p.

Then we take a look at Circle loss[5], which is:

$$L_{cir} = \log[1 + \sum_{j=1}^{L}\exp(\gamma\alpha_n^j(s_n^j - \Delta_n))\sum_{i=1}^{K}\exp(-\gamma\alpha_p^i(s_p^i - \Delta_p))]$$

where s_n means negtive similarity and s_p means positive. And in the class-level style, there is only one s_p so the loss can be shown as:

$$L_{cir} = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{\gamma\alpha_p^{y_i}(s_p^{y_i}-\Delta_p)}}{e^{\gamma\alpha_p^{y_i}(s_p^{y_i}-\Delta_p)}+\sum_{j=1,j\neq y_i}^{n}e^{\gamma\alpha_n^j(s_n^j-\Delta_n)}}$$

subject to

$$\begin{cases}\alpha_p^i = \left|O_p - s_p^i\right|_+ \\ \alpha_n^j = \left|s_n^j - O_n\right|_+\end{cases}$$

$$O_p = 1+m, O_n = -m, \Delta_p = 1-m, \Delta_n = m$$

$$W_j = \frac{W_j}{\|W_j\|}, x_i = \frac{x_i}{\|x_i\|}, s^j = W_j^T x_i$$

Circle-loss provide the self-weighted for s_n and s_p. We can also analyze that the loss is Monotonically increasing to the s_n and Monotonically decreasing to s_p, while both s_p and s_n are in (0,1). From the angel view, as shown in Fig.1, it's convergence target is to maximize θ_n to $\pi/2$ and to minimize θ_p to 0.

Based on the previous analysis, we can get two insights on improving the loss fuction.

- **Combination of advantages.** Since arc-loss provide an additive angular margin and circle-loss provide self-weighted in training, we can make a combination for these two loss to use both of their advantages.
- **Convergence target shift.** From the previous anylasis, the convergence target of circle-loss is to maximize θ_n to $\pi/2$ and the convergence target of arc-loss is even maximize θ_n to π. But in fact, we don't alwayas need to maximize θ_n to $\pi/2$ or π. Since in face rocognition situation, we can't make sure that people in different sub ids are not similar at all, it's usual that two different people have some simliarity, like 0.3 or 0.2, and try to minimise this simliarity may make model pay useless attention on easy negtive samples.To solve this problem, we give a shift on the convergence target for negtive and make easy negtive samples contribute less to the final grad.

So we proposed an asymmetric-arc-loss. The asymmetric-arc-loss can be shown like this:

$$L_{asymmetric-arc} = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{\gamma\alpha_p^{y_i}\cos(\theta_p^{y_i}+\Delta_p)}}{e^{\gamma\alpha_p^{y_i}\cos(\theta_p^{y_i}+\Delta_p)}+\sum_{j=1,j\neq y_i}^{n}e^{\gamma\alpha_n^j\cos(\theta_n^j+\Delta_n)}}$$

subject to

$$\begin{cases}\alpha_p^i = \left|O_p + \theta_p^i\right|_+ \\ \alpha_n^j = \left|O_n - \theta_n^j\right|_+\end{cases}$$

$$O_p = \pi - tm, O_n = tm, \Delta_p = tm, \Delta_n = \pi - tm$$

$$W_j = \frac{W_j}{\|W_j\|}, x_i = \frac{x_i}{\|x_i\|}, \cos\theta^j = W_j^T x_i$$

Where γ and tm are hyperparameters and $\theta_p^{y_i} + \Delta_p$, $\theta_n^j + \Delta_n$ are clip to (0, π).

Let's make a analysis on this loss. First, just like circle-loss, θ_n and θ_p get self-weighted based on their own value via α. Since O_n and O_p are fixed, the higher value of

θ_p, which is more difficult get higher weights and lower value of θ_n, also diffcult, get more weights.And turn to the easy samples, for positive, the weights are still kept, and for negtive samples, if $\theta_n^j > O_n$, their weights will become 0. Then we can see that this loss give a margin on θ instead of similarity, just like the arc-loss, to get an additive cosine margin. The decision boundary is achieved at

$$\gamma(\alpha_p \cos(\theta_p + \Delta_p) - \alpha_n \cos(\theta_n + \Delta_n)) = 0$$

What's more, seen from the grad, we take a look at item about θ_n, we assume that $v_n = \alpha_n \cos(\theta_n + \Delta_n) = (tm - \theta_n)\cos(\theta_n + \pi - tm)$ and $\frac{\partial v_n}{\partial \theta_n} = \cos(\theta_n - tm) - (\theta_n - tm)\sin(\theta_n - tm)$ so the loss get min value for $\cos(\theta_n - tm) - (\theta_n - tm)\sin(\theta_n - tm) = 0$, in our hyperparameter setting where tm = 0.65π , the θ_n is at about 0.38π, and this target can shift base on the value of tm.so this loss can focus less on easy negtive samples since their grad are smaller.

Fig 1 shows the different convergence target between these loss, and we can find that our loss's negtive convergence target shift to left, compared to arc-loss and circle-loss. We made a ablation experiment for arc-loss, circle-loss and our asymmetric-arc-loss on validation set and the results are at TABLE II.

In the actual instruction, we make a data preprosses on the training set. We use a open-source retina-face model [6] [7] [8] to make a face-detection and get landmarks, then we align the face using standard 5-point landmarks and get a 112*112 size aligned face.And then we train a general model, the details are as followed:

• **Dataset.** We use MS1M-ArcFace (85K ids/5.8M images) [14] and our self-owned face(10K ids/0.5M images) dataset for this stage.

• **Backbone.** We've tried on Mobilefacenet[9], ResNet50, ResNet101 [10] and ResNeSt [11], and finally choose ResNet101 as the backbone. TABLE I shows the perfomance of these backbone.

• **Loss.** Just as mentioned at section 2, we proposed asymmetric-arc-loss for training, and the hyperparameter setting is 0.65π for tm and 64 for γ.

• **Training settings.** We train the general model using 4 Tesla-v100 gpus at batch-size 1600, the starting learning rate is 0.1 and then decreasing to 0.01 after 100000steps then decreasing to 0.001 at 160000 steps and finally decreasing to 0.0001 at 200000steps. We use fp16 data-fomat in training to speed-up the training and maximize the batch-size.

2) Step2.Finetune model: In this step, we use the model from step1 as pretrained so the backbone is same.We use provided fairface training dataset for this stage. And we use asymmetric-arc-loss and we set same hyperparameters. At the begining of the finetuing, we freeze all layers but the last fc-layer for three epochs because the grad generated at the begining are noisy to other layers. And after three epochs, we train all layers. The learning rate is set to 0.002 for finetuing and decreas to 0.0002 after 3000 steps since we start to train all layers.

3) Step3.Most-discriminated protected-attributes data finetune: This step is quite important for the bias mitigation and easy to understand. Training a model on one attribute data can directly improve its performance on this attribute data and therefore dismiss bias. But this step needs careful tuning because the model will overfit on this attributes and lead to decreasing in acc and incresing in bias.

In this step, first we make a prediction on fairface training set using model from step2, and then choose the most-discriminated protected-attributes according to the prediction accuracy. After we get the most-discriminated protected-attributes, we finetune the model from step2 using the data with most-discriminated protected-attributes. We train all layers directly and the loss and batch-size are same with step2, but we set the learning rate wo 0.0002 at the begining and only trian for 1 epoch. The performance improvement of these finetuing steps are shown in TABLE III and TABLE IV

4) Step4.Hard-sample pick and finetune another: After we get a final finetuned model at step3, there must be some data on the fairface training set that the model can't truely predict. These are obvious hard samples. According to the existing conclusions, pay much attention to most-hard smaples will lead to bad performance , for example, in triplet loss traning, we select semi-hard sampels. But the hard-samples problem also needs to be solved, so we proposed a model fusion strategy. We get the false-predicted ids from the model, which means the predicted argmax id is not equal to the annotation id, and then finetune a model from step1 general model with just picked ids. We then get a model which performs better for those hard-sample but worse in general cases, so at the fusion step, we only take the result with extremely high confidence from the hard-sample model.

In this step, we make a prediction on training set using model from step3, and get false samples, whose prediction argmax is different from the annotation. And then we choose ids that contain these samples as a new training dataset. After we get the dataset, we finetune the model from step1 using the same settings with step2 on the choosen dataset.Finally, we get a model performance better on hard-sample and prepare this model for next model fusion.

5) Step5.Postprocess: We do three post process steps from the original result.

• **Reranking.** It's widely used in person-reid task [13] and we just use a very simple edition at this work. For template id A and B, we can get their original similarity score produced by the model, and by traversal the predictions file, we can get a set consists of k template ids which have highest similarity score to A and B, then we compare the two sets, if some template id C are both in A's top k set and B's top k set, we add a similarity score to another similarity score betweend A and B called top-k similarity. And the final similarity of A and B is a weighted sum of original similarity and top-k similarity.

First we build a top20-similar set for every template and then for a given pair A and B , we compute the overlap of A's and B's top20 dict and get the top-20 overlap similiraty.The persudo code can be shown as :

```
topkSim = 0
for item in topDict[A]:
  if(item in topDict[B]):
    topkSim += topDict[A][item] *
        topDict[B][item]
finalSimilarity = 0.65 *
    originalSimilarity + topkSim / 20 *
    0.35
```

• **BoundaryCut.** For some template ids in the predictions, there is a obvious boundary between the positive samples and negtive samples, so, we increase the similarity score up the boundary and decrease the similarity score under the boundary. For a template id A, we first traversal the predictions file, and find all pairs that contain A as a itemlist, and found its boundary. We sort the itemlist and get itemlist[1] - itemlist[i+1] as grad, and find the lowest boundary. And then increase the similarity score if it is greater than the boundary and decrease it if it s less then the boundary, the persudo code can be show as :

First we find the Boundary

```
  for item in SimDict:
    itemlist = SimDict[item]
    itemlist.sort()
    itemlist.reverse()
    SimDict[item] = (0.0, 0.0)
    for i in range(len(itemlist) - 1):
      grad = itemlist[i] - itemlist[i +
          1]
      if(grad > 0.1):
        SimDict[item] = (itemlist[i+1],
            grad)
```

And then make cut on prediction

```
thres1, grad1 = SimDict[id1]
thres2, grad2 = SimDict[id2]
finalDis = featureDis
if(featureDis > thres1 + 1e-3):
  finalDis += grad1 * 0.1
else:
  finalDis -= grad1 * 0.1
if(featureDis > thres2 + 1e-3):
  finalDis += grad2 * 0.1
else:
  finalDis -= grad2 * 0.1
```

• **Hard-Sample Fusion.** For a pair A and B, we get two similarity scores from step3 finetuned model and hard-sample model, and we only take the hard-sample model when it has extremely high confidence, the persudo code can be shown as : Just as metioned before, we train another model from hard-sample only and take its result on when it has extremely high confidence

```
if(hardSampleScore > 0.99):
  finalScore = hardSampleScore
else:
  finalScore = generalScore
```

The performance improvement of these post-process steps are shown in TABLE III and TABLE IV

E. Challenge results and final remarks

Fill Table V with your obtained results, shown in the leaderboard of the challenge[1]. Note, if you joined the challenge in the test phase, keep the "development" row blank.

TABLE V
LEADERBOARD: RESULTS OBTAINED BY THE PROPOSED APPROACH.

Phase	Rank	Bias positive pairs	Bias negative pairs	Accuracy
Development				
Test	1	0.000012	0.000059	0.999966

III. ADDITIONAL METHOD DETAILS

Please reply if your challenge entry considered (or not) the following strategies and provide a brief explanation.

- **Did you use pre-trained models?** ( ) Yes, (✓) No
  If yes, please detail:
- **Did you use external data?** (✓) Yes, ( ) No
  If yes, please detail:
  We use MS1M-ArcFace (85K ids/5.8M images) [14] and our self-owned face(10K ids/0.5M images) dataset for step1 general training.
- **Did you use other regularization strategies/terms?** ( ) Yes, (✓) No
  If yes, please detail:
- **Did you use handcrafted features?** ( ) Yes, (✓) No
  If yes, please detail:
- **Did you use any face detection, alignment or segmentation strategy?** (✓) Yes, ( ) No
  If yes, please detail:
  We use a open source retina-face detect and landmark model and use the landmarks to align the face
- **Did you use ensemble models?** (✓) Yes, ( ) No
  If yes, please detail:
  We train a general model and a hard-sample focus model and make a result fusion, since the generalization of general model is much better than hard-sample focus model, we only take the hard-sample focus model's result when it's confidence is greater than the threshold, in this work, we choose 0.99 as threshold, and there are

[1] https://competitions.codalab.org/competitions/24123

nearly no false-positive at this confidence, this fusion strategy increase our auc and decrease positive bias

- **Did you use different models for different protected groups?** ( ) Yes, (✓) No
  If yes, please detail:

- **Did you explicitly classify the legitimate attributes?** ( ) Yes, (✓) No
  If yes, please detail:

- **Did you explicitly classify other attributes (e.g. image quality)?** ( ) Yes, (✓) No
  If yes, please detail:

- **Did you use any pre-processing bias mitigation technique (e.g. rebalancing training data)?** ( ) Yes, (✓) No
  If yes, please detail:

- **Did you use any in-processing bias mitigation technique (e.g. bias aware loss function)?** (✓) Yes, ( ) No
  If yes, please detail:
  Our asymmetric-arc-loss is self-weighted so it can give more weight to samples which achieve worse performance, and bias is decreased based on greater weight on these samples.
- **Did you use any post-processing bias mitigation technique?** (✓) Yes, ( ) No
  If yes, please detail:
  We use a protected-attribute based random noise at development phase and it can decrease the bias as whatever you want. In test phase, this method is not used since bias is low enough. But from research view, this is what we thought as another viewpoint to bias and acc. For example, if we know that this model performs better on attribute A, but worse on B, just for fair purpose, making this model perform better on B is equal to making this model perform worse on A, but making this model perform worse on A is quite easy. Random noise can be thought as a method, it can be done by follow steps: first we train a classification model for two attributes, and a face-recognition model, then we evaluate the face-recognition model on the real-used domain test dataset, and we can get the bias between two attributes. Assume we perfom better on A, and in actual using we can make a attribute classifition and choose relatively sure result, for example, confidence-score is greater than 0.95 and set a probability to modify the oringal face-recognition result opposite and dynamic adjust the probability based on the performance in actul using.We can get a absolutely fair system by this way.So, if we get a model with high accuracy, it's easy to make it fair to different groups. the persudo code for this instruction is :

```
if(attibute == A):
  randomScore = random.random()
  if(randomScore < probability and
       pairScore > posiThres):
    pairScore = 1 - pairScore
  elif(andomScore < probability and
       pairScore < negThres):
    pairScore = 1 - pairScore
    #assume pairScore is normlized to
         0-1
```

IV. Code Repository

Link to a code repository with complete and detailed instructions so that the results obtained on Codalab can be reproduced locally. This includes a list of requirements, pre-trained models, and so on. Note, training code with instructions is also required. This is recommended for all participants and mandatory for winners to claim their prize. **Organizers strongly encourage the use of docker to facilitate reproducibility**.

Code repository: `https://github.com/paranoidai/Fairface-Recognition-Solution`

References

[1] ChaLearnLAP. ECCV 2020 ChaLearn Fair Face Recognition Challenge. [Online]. Available: http://chalearnlap.cvc.uab.es/challenge/38/description/

[2] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[3] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] J. Deng, J. Guo, X. Niannan, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *CVPR*, 2019.

[5] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] J. Deng, J. Guo, Z. Yuxiang, J. Yu, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-stage dense face localisation in the wild," in *arxiv*, 2019.

[7] J. Guo, J. Deng, N. Xue, and S. Zafeiriou, "Stacked dense u-nets with dual transformers for robust face alignment," in *BMVC*, 2018.

[8] J. Deng, A. Roussos, G. Chrysos, E. Ververas, I. Kotsia, J. Shen, and S. Zafeiriou, "The menpo benchmark for multi-pose 2d and 3d facial landmark localisation and tracking," *IJCV*, 2018.

[9] S. Chen, Y. Liu, X. Gao, and Z. Han, "Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices," 2018.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," 2020.

[12] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[13] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[14] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," 2016.