GICOFACE: GLOBAL INFORMATION-BASED COSINE OPTIMAL LOSS FOR DEEP FACE RECOGNITION

Xin Wei, Hui Wang, Bryan Scotney, and Huan Wan

School of Computing, Ulster University, BT370QB UK

ABSTRACT

Loss function plays an important role in CNNs. However, the recent loss functions either do not apply weight and feature normalisation or do not explicitly follow the two targets of improving discriminative ability: minimising intra-class variance and maximising inter-class variance. Besides, all of them consider only the feedback information from the current mini-batch instead of the distribution information from the whole training set. In this paper, we propose a novel loss function – Global Information-based Cosine Optimal loss (Gico loss). Gico loss is applied with weight and feature normalisation, designed explicitly following the aforementioned two targets of improving discriminative ability, and is guided by the distribution information from the whole training set. Extensive experiments are conducted on multiple public datasets, which confirms the effectiveness of the proposed Gico loss and shows that we achieve state-of-the-art performance.

Index Terms— Face recognition, Deep learning, CNNs, Loss function, Discriminative ability.

1. INTRODUCTION

Convolutional neural networks (CNNs) have demonstrated impressive performance on face recognition, where the loss function plays an important role in this process. To learn highly discriminative features, many different loss functions are proposed in recent years [1–9]. Currently, the best performing loss functions in face recognition can be divided into two types – the loss functions based on Euclidean distance [1–5] and the loss functions based on cosine similarity [6–9].

Typical Euclidean distance-based losses include Centre loss [3], Marginal loss [4] and Range loss [5]. All of them add another penalty to implement the joint supervision with softmax loss and are designed based on the following two targets: minimising intra-class variance and maximising inter-class variance. From the relevant experimental results of the methods above [1–5], it can be found that both two targets contribute to performance. The loss functions based on Cosine similarity include L-Softmax loss [6], A-Softmax loss [7] and AM-Softmax loss [8]. They are derived from softmax loss by adding additional margin constraints. The experiments in [7] demonstrate that L2 weight normalisation improves the performance, though the improvement is very limited. Feature normalisation brings advantages including better performance and better geometrical interpretation, which are revealed in [10–13].

Table 1. Properties of different losses in deep face recognition.

	Optimise Intra-class Variance	Optimise Inter-class Variance	WN	FN	Feedback Source
Contrastive loss [1]	Yes	Yes	No	No	mini-batch
Triplet loss [2]	Yes	Yes	No	No	mini-batch
Centre loss [3]	Yes	No	No	No	mini-batch
Marginal loss [4]	Yes	Yes	No	No	mini-batch
Range loss [5]	Yes	Yes	No	No	mini-batch
L-Softmax loss [6]	No	Yes	No	No	mini-batch
A-Softmax loss [7]	No	Yes	Yes	No	mini-batch
AM-Softmax loss [8]	No	Yes	Yes	Yes	mini-batch
ArcFace loss [9]	No	Yes	Yes	Yes	mini-batch
Gico loss*	Yes	Yes	Yes	Yes	global info

[1] WN: weight normalisation. FN: feature normalisation.

Table 1 summarises the properties of the most recent and best performing loss functions. However, these loss functions either do not apply weight and feature normalisation like Contrastive loss, Triplet loss, Centre loss, Range loss and Marginal loss; or do not explicitly follow the two targets of improving discriminative ability, like L-Softmax loss, A-Softmax loss, AM-Softmax loss and ArcFace. In this paper, we propose a new loss function, namely Global Information-based Cosine Optimal loss (Gico loss). The deep model trained with Gico loss is named GicoFace. The relevant properties of Gico loss are also shown in Table 1 where Gico loss possesses all four properties of optimising intra-class and inter-class variance, and weight and feature normalisation. Different from the other losses, Gico loss is guided by the distribution information from the whole training set.

The main contributions of this paper are summarised as follows: (a). We propose a novel loss function to improve the discriminative ability of the deep features. To the best of our knowledge, it is the first loss that simultaneously satisfies all the first four properties in Table 1 and also the first attempt to use global information as the feedback information. (b). We propose and implement three different versions of Gico loss and analyse their performance variation on multiple datasets.

To develop Gico loss, we propose an algorithm to learn the cosine similarity between the class centre and the class edge. (c). We conduct extensive experiments on multiple public benchmark datasets including LFW [14], SLLFW [15] and YTF [16] datasets. Experimental results presented in Section 3 demonstrate the state-of-the-art performance of GicoFace.

2. FROM SOFTMAX LOSS TO GICO LOSS

2.1. Softmax Loss and Centre Loss

Softmax loss is the most commonly used loss function in deep learning, which can be formulated as:

$$\mathcal{L}_S = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{W_{y_i}^T f_i + b_{y_i}}}{\sum_{j=1}^{P} e^{W_j^T f_i + b_j}} \tag{1}$$

where N denotes batch size, P represents the number of classes in the whole training set, $f_i \in R^d$ is the feature vector of the ith sample belonging to the y_ith class, $W_j \in R^d$ is the jth column of the weight matrix W in the final fully connected layer and b_j is the bias term of the jth class[1]. From Eq(1), we can see that Softmax loss is essentially the cross-entropy between the predicted label and the true label, which means Softmax loss focuses only on the correctness of classification. In other words, Softmax loss aims at separating the samples of different classes instead of learning discriminative features and enlarging the margin between neighbour classes. This approach is appropriate for closed-set tasks, like most cases in object recognition and behaviour recognition, where all the testing classes are predefined in the training set. However, most of the application scenarios of face recognition are open-set tasks, as it is almost impossible to collect all the faces that may appear in the test stage. To improve the discriminative ability of the features, Wen et al. [3] proposed Centre loss to minimize the intra-class distance whose formulation is shown below:

$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{N} ||f_i - c_{y_i}||_2^2 \tag{2}$$

where c_{y_i} denotes the class centre of the y_ith class. Centre loss is the sum of all the distances between each sample and its class centre. Centre loss is used in conjunction with Softmax loss:

$$\mathcal{L} = \mathcal{L}_S + \lambda\mathcal{L}_C \tag{3}$$

where λ is the hyper-parameter for adjusting the impact of these two losses. Centre loss optimises only the intra-class variance and it doesn't apply weight and feature normalisation.

2.2. Variants of Softmax Loss based on Cosine Similarity

L-Softmax loss, A-Softmax loss, AM-Softmax loss and ArcFace loss are variants of Softmax loss based on cosine similarity. All of them are derived from the original Softmax loss in Eq.(1), replacing the distance measurement from Euclidean distance to cosine similarity. They transform the FC layer formulation from $W_{y_i}^T f_i + b_{y_i}$ to $\|W_{y_i}\|\|f_i\|cos\theta_{y_i}$ by setting the bias b_{y_i} to 0, where θ_{y_i} is the angle between W_{y_i} and f_i. However, they have different choices for weight and feature normalisation, and use different ways to add marginal constraints.

In L-Softmax loss and A-Softmax loss, the marginal constraints are added by a multiplier m on the angle, namely replacing $cos\theta_{y_i}$ with $cos(m\theta_{y_i})$. So we say that L-Softmax loss and A-Softmax loss apply the multiplicative angular margin. Different from L-Softmax loss, weight normalisation is introduced in A-Softmax loss, which sets $\|W_{y_i}\| = 1$ by L2 normalisation. AM-Softmax loss further applies feature normalisation and replaces the multiplicative angular margin by the additive cosine margin, which is formulated as:

$$\mathcal{L}_{AM} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(cos(\theta_{y_i})-m)}}{e^{s(cos(\theta_{y_i})-m)} + \sum_{j=1,j\neq y_i}^{P} e^{scos(\theta_j)}} \tag{4}$$

where $\|f_i\|$ is fixed by L2 normalisation and is re-scaled to s. So $\|f_i\|$ is replaced with s in Eq.(4). Based on AM-Softmax loss, ArcFace loss further replaces $cos(\theta_{y_i})-m$ with $cos(\theta_{y_i}+m)$ enabling m to have better geometric meaning.

2.3. The Proposed Gico Loss

In this section we propose a new loss function to integrate the advantages of existing losses with some important new properties. Firstly, we apply L2 weight normalisation by fixing $b_j = 0$ and $||W_j|| = 1$. We also apply L2 normalisation on the feature vector f_i and re-scale $\|f_i\|$ to s. Similar to Centre loss, Gico loss is used in conjunction with AM-Softmax loss. Here we do not adopt Softmax loss like Centre loss, because AM-Softmax loss shows slightly better performance than Softmax loss. The total loss is thus:

$$\mathcal{L} = \mathcal{L}_{AM} + \lambda\mathcal{L}_G \tag{5}$$

In designing the Gico loss, two aspects are considered: minimising the intra-class variance and maximising the inter-class variance. These two aspects correspond to two "lite" versions of Gico loss, respectively. Finally, we construct a standard version of Gico loss which is the combination of the two lite versions. To minimise the intra-class variance, we propose a "lite" version of Gico loss (Gico Lite A), which is formulated as below:

$$\begin{gathered}\mathcal{L}_{G_A} = \frac{P}{\sum_{j=1}^{P}\frac{R(j)+1}{2}} \\ R(j) = cos(c_j, e_j)\end{gathered} \tag{6}$$

where P is the number of classes in the whole training set, c_j is the centre of class j, and e_j denotes the edge of class j (i.e. the farthest sample of class j). $R(j)$ represents the cosine range of class j, namely the cosine similarity between the class centre and the edge of class j. During the training, the deep features are changing after each mini-batch, which also leads to the change of c_j and e_j. Ideally, c_j and e_j should be calculated by traversing the entire training set and should be updated after each mini-batch. However, this would require

[1] To save space, we just define the mathematical symbols once and then use them following the initial definition without additional explanation.

massive computing power that is completely impractical for the existing hardware. Currently, a deep neural network is trained by iteratively updating the network parameters based on the feedback information from each mini-batch. This is a practical solution due to two constraints: the computing power and the memory size of GPU, TPU or other similar processing units. Without the computing power constraint, the deep neural network could be trained with the entire training set as the source of feedback information and would directly optimise the sample distribution of the entire training set. Without the memory size constraint, the deep neural network would input the entire training set into the memory instead of processing the data mini-batch by mini-batch. Perhaps just because of the above two constraints, there is no loss that uses the entire dataset as the source of feedback information to optimise the CNNs in face recognition.

Here we break through the first constraint by two approximate solutions. From Eq(4), we can see that the key optimisation object of the AM-Softmax loss is actually minimising θ_{y_i} while maximising θ_j. θ_{y_i} is the angle between W_{y_i} and f_i. θ_j is the angle between W_j and f_i where $j \neq y_i$. In other words, AM-Softmax loss tries to reduce the distances between W_j and the sample features in the jth class ($j = 1, 2, ..., P$). As the training goes on, W_j is automatically optimised to the centre of class j ($j = 1, 2, ..., P$), because this leads to the minimum distance sum between W_j and the sample features in the jth class. Therefore, we can simply use W_j as the substitution of c_j, which does not require any additional computing power. For e_j and $R(j)$, we propose a learning algorithm to recursively update the range of each class. At the beginning, $R(j)$ is initialised to 1. Then we update $R(j)$ using the following iterations:

$$R(j)^{t+1} = R(j)^t + \sum_{i=1}^{N} \phi(y_i, j) \cdot \Delta R_i, j = 1, 2, ..., P. \quad (7)$$

$$\Delta R_i = \begin{cases} cos(W_{y_i}, f_i) - R(y_i)^t, & R(y_i)^t > cos(W_{y_i}, f_i) \\ \beta \cdot (cos(W_{y_i}, f_i) - R(y_i)^t), & R(y_i)^t \leq cos(W_{y_i}, f_i) \end{cases} \quad (8)$$

where $\phi(y_i, j) = 1$ when $y_i = j$, otherwise $\phi(y_i, j) = 0$. β is the shrink rate which is used to adjust the shrink speed of the learned class range. The basic idea of the learning algorithm includes two cases: (a). if the cosine similarity between the input sample and its corresponding class centre is smaller than the recorded class range, replace the class range directly with their cosine similarity; (b). on the contrary, let the class range shrink by scaling their cosine similarities with β. Case (a) keeps the learned class range up to date. However, as the training goes on, the real class range will become smaller and smaller. So case (b) is used to help the learned class range shrink to the real value.

To maximise the inter-class variance, we also propose another "lite" version of Gico loss (Gico Lite B):

$$\begin{gathered} \mathcal{L}_{G_B} = \frac{\sum_{Top}(A, K)}{K} \\ A = \{\frac{cos(W_a, W_b) + 1}{2} : a, b = 1, 2, 3, ..., P; a > b\} \end{gathered} \quad (9)$$

where $\sum_{Top}(A, K)$ denotes the sum of the K largest elements in set A. The purpose of Gico Lite B is to find K pairs of nearest class centres in the entire training set and to calculate the sum of their distances. Compared with the non-adjacent class centres, the corresponding classes of the adjacent centres have a high probability to have small margins or have overlaps. If all adjacent classes have proper margins, the non-adjacent classes would have larger margins. Therefore, it is not necessary to take all centre pairs into account. The most effective way is to optimise the distances of all the adjacent centres. However, it is time-consuming to calculate how many adjacent centre pairs exist in the hypersphere. Here we adopt a conservative strategy, namely set the value of K to P where P is the number of classes. Because the minimum number of adjacent centre pairs is P which happens when all the class centres line up in a circle on the surface of the hypersphere.

To achieve the best performance we integrate the above two lite versions to create the standard version of Gico loss (Gico Std):

$$\mathcal{L}_{G_{std}} = L_{G_A} * L_{G_B} = \frac{P * \sum_{Top}(A, K)}{K * \sum_{j=1}^{P} \frac{R(j)+1}{2}} \quad (10)$$

3. EXPERIMENTS

3.1. Experiment Settings

Our network models are implemented by Tensorflow[2] with Inception-ResNet-v1 [17] as the trunk network. We combine Inception-ResNet-v1 with different losses resulting in 5 different combinations: (1). ResNet+Softmax, (2). ResNet+AM-Softmax, (3). ResNet+Gico Lite A, (4). ResNet +Gico Lite B, and (5). ResNet+Gico Std. In all experiments, we set 320 as the epoch size, 120 as the batch size, 5e-4 as the weight decay, 0.4 as the keep probability of the fully connected layer, 512 as the embedding size and 0.01 as the shrink rate. We manually optimise the hyper parameter λ. Since it is not very sensitive to the performance, we just try multiple different values on each testing set and choose the value that leads to the best result. The initial learning rate is set to 0.05 and is reduced by a factor of 10 every 100,000 iterations.

VGGFace2 [18] is the training data in our experiments. To guarantee the reliability of the results, we removed the identities which might be overlapped with the testing sets from VGGFace2, but we did not do data cleaning as VGGFace2 is a very clean dataset. Finally, the preprocessed training set contains 3.05 million face images. We applied the same preprocessing pipeline on the training set and the testing sets. Firstly MTCNN [19] is employed for face detection. MTCNN occasionally fails to detect the face. If this occurs for a training image, the image is simply abandoned. If it occurs for

[2] https://www.tensorflow.org/

Table 2. Verification performance of state-of-the-art methods on LFW and YTF datasets.

Methods	Images	LFW(%)	YTF(%)
ICCV17' Range Loss [5]	1.5M	99.52	93.7
CVPR15' DeepID2+ [20]		99.47	93.2
CVPR14' Deep Face [21]	4M	97.35	91.4
CVPR15' Fusion [22]	500M	98.37	
ICCV15' FaceNet [2]	200M	99.63	95.1
ECCV16' Centre Loss [3]	0.7M	99.28	94.9
NIPS16' Multibatch [23]	2.6M	98.20	
ECCV16' Aug [24]	0.5M	98.06	
ICML16' L-Softmax [6]	0.5M	98.71	
CVPR17' A-Softmax [7]	0.5M	99.42	95.0
Softmax	3.05M	99.50	95.22
AM-Softmax	3.05M	99.57	95.62
Gico Lite A	3.05M	99.60	95.70
Gico Lite B	3.05M	99.62	95.78
Gico Std*	3.05M	99.63	95.82

Table 3. Verification performance of different methods on SLLFW.

Method	Images	LFW(%)	SLLFW(%)
Deep Face [21]	0.5M	92.87	78.78
DeepID2 [1]	0.2M	95.00	78.25
VGG Face [25]	2.6M	96.70	85.78
DCMN [26]	0.5M	98.03	91.00
Noisy Softmax [27]	0.5M	99.18	94.50
Softmax	3.05M	99.50	96.17
AM-Softmax	3.05M	99.57	98.02
Gico Lite A	3.05M	99.60	98.15
Gico Lite B	3.05M	99.62	98.13
Gico Std*	3.05M	99.63	98.17

a testing image, we use the provided official landmarks or bounding boxes instead. All the face images are cropped to the size of 160*160. Random horizontal flipping is performed on the training image to enhance the randomness of the training data. The final features of a testing image are generated by concatenating the features of the original image and the features of its horizontally flipped counterpart so as to improve the recognition accuracy.

3.2. Results on LFW, YTF and SLLFW

In this section we compare the proposed methods with the state-of-the-art methods on LFW, YTF and SLLFW. LFW [14], collected from the web, contains 13,233 face images with large variations in facial paraphernalia, pose and expression. Following the standard experimental protocol of "unrestricted with labelled outside data" [28], we test 6,000 face pairs according to the given pair list. YTF [16] contains 3,425 videos obtained from YouTube. We follow the standard experimental protocol of "unrestricted with labelled outside data" [16] to do the evaluation on the given 5,000 video pairs.

Table 2 shows the results of the proposed methods and the state-of-the-art methods[3] on LFW and YTF. The results of the benchmark methods shown in the upper part of the table are cited from their original papers. From Table 2, we can observe the following. Gico Std shows higher verification accuracy on LFW than Softmax, AM-Softmax, Gico Lite A and Gico Lite B. Gico Std ties with FaceNet for first place on LFW. However FaceNet uses 200 million images for training, whilst Gico Std uses only 3.05 million images. Gico Std also beats the other benchmarks methods on LFW, most of which are published in leading computer vision conferences. On YTF dataset the proposed Gico loss still has better performance than the other benchmark methods, which demonstrates the state-of-the-art performance of the Gico loss.

LFW is a popular face dataset. But more and more methods are gradually touching its theoretic upper limit. Consequently, it becomes more and more difficult to differentiate different methods on LFW. To confirm the performance of the proposed methods, we conducted an additional experiment on SLLFW [15]. SLLFW uses the same positive pairs as LFW for testing, but in SLLFW, 3000 similar-looking face pairs are deliberately selected out from LFW by human crowdsourcing to replace the random negative pairs in LFW. SLLFW adds more challenges to the testing, causing the accuracy of the same state-of-the-art methods drops about 10-20%.

Table 3 shows the verification accuracy of different methods on SLLFW. The results of some benchmark methods are shown in the top half of the table. These results are publicly accessible[4] and provided by the SLLFW team [26]. As can be seen from Table 3, Gico loss achieves considerably better performance than other methods on SLLFW. In the top half of the table, the accuracy of the benchmark methods drops by between 16.75% and 4.68% from LFW to SLLFW. By comparison, the accuracy of Gico loss drops by between 1.45% and 1.49%. The results on SLLFW further confirm the performance of the proposed methods. More experimental results on other datasets can be found in [29].

4. CONCLUSION

In this paper, we present a novel loss function – Global Information-based Cosine Optimal loss (Gico loss). Gico loss integrates the advantages of the best losses proposed in recent years in face recognition. To the best of our knowledge, Gico loss is the first attempt to use global information as the feedback in face recognition. To make Gico loss possible, we propose a novel algorithm to learn the cosine similarity between the class centre and the class edge. Extensive experiments are conducted on LFW, SLLFW and YTF datasets. Results demonstrate the effectiveness of the proposed Gico loss and show that we achieve state-of-the-art performance.

[3] ArcFace [9] is not included in Table 2 as it hasn't been formally published, which means it hasn't passed the peer review. Additionally, after reviewing the comments on the public codes of Arcface, we found that the experimental methods and the results are still controversial.

[4] http://www.whdeng.cn/SLLFW/index.html#reference

5. REFERENCES

[1] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in *NIPS*, 2014, pp. 1988–1996.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *CVPR*, 2015, pp. 815–823.

[3] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A Discriminative Feature Learning Approach for Deep Face Recognition,” in *ECCV*, 2016, pp. 499–515.

[4] J. Deng, Y. Zhou, and S. Zafeiriou, “Marginal loss for deep face recognition,” in *CVPR*, vol. 4, no. 6, 2017.

[5] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, “Range Loss for Deep Face Recognition with Long-Tailed Training Data,” in *ICCV*, 2017, pp. 5419–5428.

[6] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-Margin Softmax Loss for Convolutional Neural Networks,” in *ICML*, 2016, pp. 507–516.

[7] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “SphereFace: Deep Hypersphere Embedding for Face Recognition,” in *CVPR*, 2017, pp. 6738–6746.

[8] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive Margin Softmax for Face Verification,” *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[9] J. Deng, J. Guo, and S. Zafeiriou, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition,” *arXiv:1801.07698 [cs]*, 2018, arXiv: 1801.07698.

[10] 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,” *arXiv:1801.09414 [cs]*, 2018, arXiv: 1801.09414.

[11] R. Ranjan, C. D. Castillo, and R. Chellappa, “L2-constrained softmax loss for discriminative face verification,” *arXiv preprint arXiv:1703.09507*, 2017.

[12] W. Liu, Y.-M. Zhang, X. Li, Z. Yu, B. Dai, T. Zhao, and L. Song, “Deep hyperspherical learning,” in *NIPS*, 2017, pp. 3950–3960.

[13] Y. Liu, H. Li, and X. Wang, “Rethinking feature discrimination and polymerization for large-scale recognition,” *arXiv preprint arXiv:1710.00870*, 2017.

[14] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” University of Massachusetts, Tech. Rep. 07-49, 2007.

[15] N. Zhang and W. Deng, “Fine-grained lfw database,” in *International Conference on Biometrics*, 2016, pp. 1–6.

[16] L. Wolf, T. Hassner, and I. Maoz, “Face recognition in unconstrained videos with matched background similarity,” in *CVPR*, 2011, pp. 529–534.

[17] C. Szegedy, S. Ioffe, and V. Vanhoucke, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *CoRR*, vol. abs/1602.07261, 2016.

[18] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “Vggface2: A dataset for recognising faces across pose and age,” *arXiv preprint arXiv:1710.08092*, 2017.

[19] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[20] Y. Sun, X. Wang, and X. Tang, “Deeply learned face representations are sparse, selective, and robust,” in *CVPR*, 2015, pp. 2892–2900.

[21] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” in *CVPR*, 2014, pp. 1701–1708.

[22] ——, “Web-scale training for face identification,” in *CVPR*, 2015, pp. 2746–2754.

[23] O. Tadmor, T. Rosenwein, S. Shalev-Shwartz, Y. Wexler, and A. Shashua, “Learning a Metric Embedding for Face Recognition Using the Multibatch Method,” in *NIPS*, USA, 2016, pp. 1396–1397.

[24] I. Masi, A. T. Tran, T. Hassner, J. T. Leksut, and G. Medioni, “Do We Really Need to Collect Millions of Faces for Effective Face Recognition?” in *ECCV*, 2016, pp. 579–596.

[25] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep Face Recognition.” in *BMVC*, vol. 1, 2015, p. 6.

[26] W. Deng, J. Hu, N. Zhang, B. Chen, and J. Guo, “Fine-grained face verification: Fglfw database, baselines, and human-dcmn partnership,” *Pattern Recognition*, vol. 66, pp. 63–73, 2017.

[27] B. Chen, W. Deng, and J. Du, “Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation,” in *CVPR*, 2017.

[28] G. B. Huang and E. Learned-Miller, “Labeled faces in the wild: Updates and new reporting procedures,” *University of Massachusetts, Amherst, Tech. Rep*, pp. 14–003, 2014.

[29] [Online]. Available: https://drive.google.com/open?id=11V2bmAY-A4_Xsch-zyHpOLxytTn6yx0v