Deep Embeddings

Anonymous ICCV submission

Paper ID ****

Abstract

Deep embedding extraction has been extremely successful in various applications, including face verification, clustering and image retrieval. Broadly speaking, it involves optimizing some variants of contrastive loss or triplet loss in a way that similar images have similar embeddings, and vice versa. While the formulations usually seem intuitive and simple, in practice for successful training, it is crucial to use various example-sampling heuristics to avoid bad local optima and to accelerate convergence.

It is important to note that it is not just the loss function, but also these heuristics together, that define the true objective function. While a great amount of (possibly sophisticated) variants of loss function have been proposed, the importance of these sampling/training procedures is largely neglected, and a principled study is missing.

In this work, we study, propose and evaluate a class of sampling/training procedures and loss functions, which are stable to train in most settings. In addition, we show that using our strategies, even the simplest contrastive loss is sufficient to achieve state-of-the-art results on multiple datasets in multiple domains.

1. Introduction

Our main contributions are as follows:

1. We propose and evaluate a class of sampling/training procedures and loss functions, which are stable to train in most settings.
2. We study/compare/analyze various deep embedding algorithms considering not just the loss functions but also their training procedures in a unified framework.
3. Extensive experiments show that using the proposed strategies, vanilla contrastive loss is sufficient to achieve state-of-the-art results.

2. Related works

The idea of using neural networks to extract features that respect certain relationships dates back to the 90s. [3] first proposed a "Siamese Network" for signature verification, and later similar ideas were used for face verification and other applications [5, 9]. Broadly speaking, the center idea is finding an embedding space such that similar examples have similar embeddings and vice versa. However, given the limited computing power and their nature of non-convexity, these approaches did not enjoy as much attention as they do today.

Others seek to achieve similar goals with convex optimization approaches [30, 20, 27, 6]. Among these, [20, 27] propose to use relative relationship between examples, which motivated later development of triplet losses.

In recent years, given the astonishing breakthroughs in deep convolutional neural networks (CNNs) [15, 23, 10], neural network-based metric learning regained popularity in computer vision community. These methods give state-of-the-art results in various areas, such as zoro-shot learning [4], visual search [8, 1], face recognition/verification [19, 18], etc.

Among these the triplet loss is extremely successful. For example, [19, 18] use triplet loss and achieved state-of-the-art performance in face verification that outperforms humans. The success encourages abundant works striving improving upon vanilla triplet losses. Some attempted to go beyond triplets or pairs, and construct loss functions that use more examples in one term. For example, [21] and [17] proposed to use all negatives in a batch for each positive pair (in contrast to one, as in triplet loss), at the cost of higher computational complexity. Similarly, [11] proposed to use quadruplets instead of triplets.

Some strive to improve hard negative mining. [32] trained an ensemble of multiple models, each of which focuses on examples of certain "hard levels." [11] designed a new network module on top of the original model with an additional "metric loss" in an attempt to learn a metric that can be used to select better hard samples.

While they are highly successful in certain settings, in practice triplet loss and their extensions are known to be unstable in training. In this paper, we propose and evaluate a class of sampling/training procedures and loss functions, which are stable to train in most settings. We show that using these guidlines, the simplest contrastive loss is sufficient to outperform all other methods.

3. Background

3.1. Loss functions

Contrastive loss

$$\ell^{(\text{contrast})}(i,j) := y_{i,j} D_{i,j}^2 + (1-y_{i,j})\left[\alpha - D_{i,j}\right]_+^2, \quad (1)$$

where $D_{i,j} := \|f(x_i) - f(x_j)\|$.

Triplet loss

$$\ell^{(\text{triple})}(a,p,n) := \left[D_{a,p}^2 - D_{a,n}^2 + \alpha\right]_+ \quad (2)$$

3.2. Traditional training procedures

From risk minimization perspective, one might aim at optimizing

$$\sum_{t \in \{\text{all tuples}\}} \ell^{(\cdot)}(t) \quad (3)$$

However, it is computationally infeasible to enumerate through $\mathcal{O}(n^2)$ or even $\mathcal{O}(n^3)$ such tuples. In addition, most of these tuples would induce small or zero losses, when the network approaches a good solution. To accelerate convergence while maintaining training stability, variants of hard negative mining, batch construction methods are performed.

Hard negative mining One common such techniques is to use only tuples that have non-zero loss. Some sample uniformly from these violating examples [18], some sample the hardest violating examples [19], and some combine the two [7]. For triplet loss training, the most popular practice is to *not* sample the hardest examples though [19, 18]. This in practice stabilizes training and tends to converge to a better solution. However, in Section **??**, we will show that this would cause slower convergence is most important/challenging examples are ignored.

Batch construction Various ways of constructing batches are also proposed as people realize that randomly sampling of tuples naively leads to inferior convergence and results. For example, in [19] a batch is constructed such that each class in batch have at least 40 images, and bach size is set to be 1800.

4. A unified view

While not as pronounced, these techniques fundamentally change the loss function.

Corrected loss functions. The importance of these techniques are evident in the following triplet-loss example. In [19], [18], and [20], their techniques induce the following three losses respectively:

$$\begin{aligned}
&\ell^{(\text{triplet, semi-hard})}(a,p) && (4)\\
&:= \max_{n: D_{a,n} > D_{a,p}, n \in \mathcal{X}_n} \left(D_{a,p}^2 - D_{a,n}^2 + \alpha\right)_+ && (5)\\
& && (6)
\end{aligned}$$

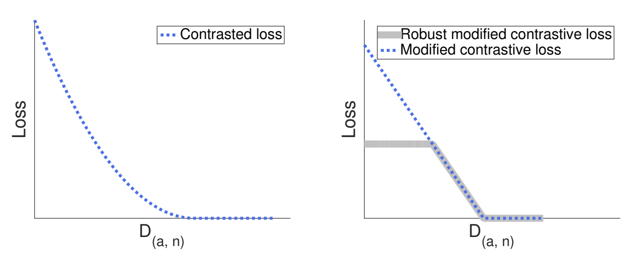
$$\begin{aligned}
&\ell^{(\text{triplet, random-violate})}(a,p) && (7)\\
&:= \sum_{n: D_{a,n} \in [D_{a,p}, D_{a,p}+\alpha], n \in \mathcal{X}_n} \left(D_{a,p}^2 - D_{a,n}^2 + \alpha\right)_+ && (8)\\
& && (9)
\end{aligned}$$

$$\begin{aligned}
&\ell^{(\text{triplet, random})}(a,p,n) && (10)\\
&:= \left(D_{a,p}^2 - D_{a,n}^2 + \alpha\right)_+, && (11)
\end{aligned}$$

where (a,p,n)-tuples are uniformly sampled for $\ell^{(\text{triplet, random})}$, and $\mathcal{X}_n$ is some set of negative examples. We see they are fundamentally different optimization objectives. These techniques are developed independently without mutual comparison, and the pros and cons are not clear.

Aggregated loss. In addition, tuples (a,p) are also sampled by some sampling algorithm $\mathcal{P}_{a,p}$. Thus the true objective of [19] could be formally written as optimizing on the expectation over this sampling distribution, i.e.

$$\mathbf{E}_{\mathcal{P}_{a,p}}\left[\ell^{(\text{triplet, semi-hard})}(a,p)\right] \quad (12)$$

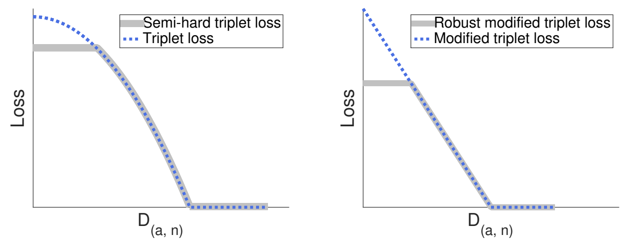


(a) Contrastive loss. (b) Modified contrastive loss.

Figure 1: Contrastive loss

5. Proposed strategies

We found that to achieve successful and fast training, three principles are important: balanced sampling, stable losses, and an aggressive yet stable hard-negative mining.



(a) Triplet loss. (b) Modified triplet loss.

Figure 2: Triplet loss

5.1. Balanced classes, batches and tuples.

A good strategy we found is to construct an objective and batches that are as balanced as possible, at all of the following three levels. First, we found that it is advantageous to sample classes uniformly instead of sampling tuples uniformly and consequently biasing towards large classes. For example, in most contrastive-loss optimizing algorithms, positive pairs are sampled uniformly across all possible pairs. Roughly speaking, in this way the training loss of a class c of size $|c|$ is effectively weighted by a factor of $\binom{|c|}{2}$. Second, we found that training is more stable when sampling the same amount of images per batch. [21] use similar techniques. Formally,

$$c_i \sim \mathcal{U}(\mathcal{C}), i = 1, \dots, B/k \tag{13}$$
$$z_{ij} \sim \mathcal{U}(\mathcal{X}_{c_i}), j = 1, \dots, k \tag{14}$$

(CY: strictly speaking, not exactly this. we sample without replacement)

Third, sampling sampling one negative for each end of a positive pair also helps stabilizing training, namely for positive pair (i, j),

$$\ell^{(\text{stable contrast})}(i, j) \tag{15}$$
$$:= (D_{i,j} - \beta + \alpha)_+ + \left(\beta - D_{i,y^\star(i)} + \alpha\right)_+ \tag{16}$$
$$\tag{17}$$

5.2. Modified contrastive/triplet loss.

Another source of instability of training is the mismatching scale of **repelling gradients**, which repels negative pairs, and **attracting gradients**, which attract positive pairs. To see this, take triplet loss for example,

$$\ell^{(\text{triple})} := \frac{1}{2}\left(\|x_a - x_p\|^2 - \|x_a - x_n\|^2 + \alpha\right)_+$$
$$\partial\ell^{(\text{triple})}/\partial x_a = x_n - x_p$$
$$\partial\ell^{(\text{triple})}/\partial x_p = x_p - x_a, \quad \text{and} \quad \partial\ell^{(\text{triple})}/\partial x_n = x_a - x_n.$$

Note that the hard negatives (the ones close to anchor) have small gradients (see Figure 2a), while the hard positives have large gradients. This causes the training to draw all examples close, and eventually converge to a bad saddle point where all examples have the same embedding. It is hard to escape from this point as at this point all three examples have gradient zero. Semi-hard negative mining was proposed to address this issue by not sampling these harder negatives, but we can simply modify the loss function to overcome this problem, i.e.

$$\ell^{(\ell_2\text{triple})} := \left(\|x_a - x_p\| - \|x_a - x_n\| + \alpha\right)_+$$
$$\partial\ell^{(\ell_2\text{triple})}/\partial x_a = \frac{x_a - x_p}{\|x_a - x_p\|} - \frac{x_a - x_n}{\|x_a - x_n\|}$$
$$\partial\ell^{(\ell_2\text{triple})}/\partial x_p = \frac{x_p - x_a}{\|x_p - x_a\|} \text{ and } \partial\ell^{(\ell_2\text{triple})}/\partial x_n = \frac{x_a - x_n}{\|x_a - x_n\|}$$

Similar issues appear for contrastive loss when the positive pairs and negative pairs have mismatching scales of gradients, and this causes instability of training. We thus propose to use a modified contrastive loss

$$\ell^{(\ell_2\text{ contrast})} := y_{i,j}\|x_i, x_j\| + (1 - y_{i,j})\left(\alpha - \|x_i, x_j\|\right)_+, \tag{18}$$

as shown in 1b.

5.3. Hard negative mining with robust losses

To achieve stable training and avoid the vanishing gradient problem mentioned in the previous subsection, instead of mining the hardest negative examples, the common practice is using the so called semi-hard negative mining [19, 18]. These semi-hard negatives are the hardest in those that are further away from anchor than the positive example (so they are not really hard). This induces

$$\ell^{(\text{triplet, semi-hard})}(a, p) \tag{19}$$
$$:= \max_{n: D_{a,n} > D_{a,p}, n \in \mathcal{X}_n} (D_{a,p} - D_{a,n} + \alpha)_+^2, \tag{20}$$

where $\mathcal{X}_n$ is some set of negative examples. [7] used a different approach that initially train with random triplets, and then hardest triplets afterwards. These techniques share similar motivations to curriculum learning [2, 13].

Note however, these methods ignore challenging and thus important examples and lead to slower convergence. We here propose the following sampling distribution such that it is stable to train yet still mines the hardest examples so that it learns efficiently. (CY: our d^{D-1} argument doesn't hold for unit-sphere.)

$$\ell^{(\text{robust hard contrast})}(i, j) \tag{21}$$
$$:= [D_{i,j} - \alpha]_+^2 + \max_{n: D_{i,n} > \beta, n \in \mathcal{X}_n} [\alpha - D_{i,n}]_+^2 \tag{22}$$

where $\mathcal{X}_n$ is some set of negative examples.

We also tried self-paced learning approaches [16] where we start with the simplest examples only, and then decrease the threshold later on to adopt harder and harder examples. They achieve similarly good performance as our proposed XX distribution, yet introduce one more parameter to control the *pace* of learning, so we did not pursue this direction.

6. Experiments

Our proposed method is very stable to train. In all of the following experiments, we use the same model parameters and sampling procedures for all datasets, and all of them achieve very competitive results.

Training of contrastive loss follows [5], where half of the training pairs are positive and half of them are randomly sampled negatives.

6.1. Data sets

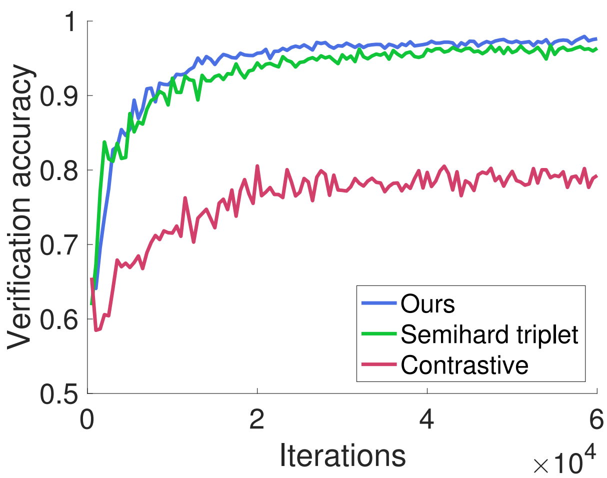


Figure 3: Convergence

Verification:

- **Labeled Faces in the Wild (LFW).**
- **Something else.**

Image retrieval:

- **CUB-200-2011.**
- **CARS196.**
- **Stanford Online Products.**

Note that CASIA face dataset is known to contain many incorrect labels[1].

[1] According to https://github.com/happynear/FaceVerification, 27,703 images in CASIA dataset have incorrect labels.

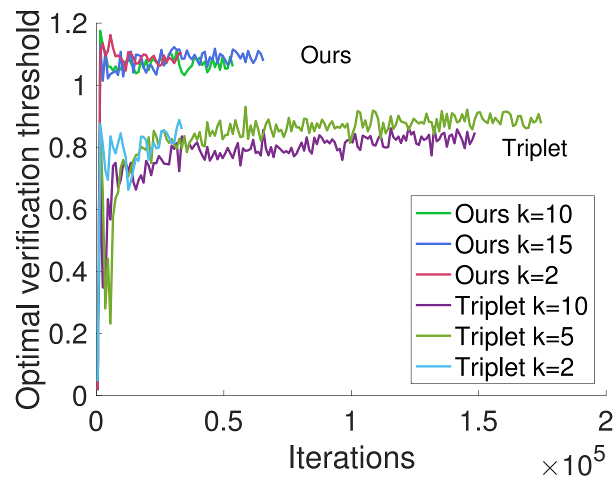


Figure 4: Triplet loss

Domain	Dataset	# classes	# images
Faces	Train: CASIA [31]	10,575	494,414
	Test: LFW [12]	5,749	13,233
Products	Stanford Online Products [17]	22,634	120,053
Cars	CARS196 [14]	196	16,185
Birds	CUB200-2011 [28]	200	11,788

Table 1: Datasets

Loss	Sampling	Accuracy	AUC	100% - EER
Triplet Semihard	random			
Triplet Semihard	$m = 2$			
Triplet Semihard	m = 5			
Triplet Semihard	k=10			
Triplet Semihard	k=20			
Ours	random			
Ours	k=2			
Ours	k=5			
Ours	k=10			
Ours	k=20			

Table 2: Ablation study on LFW.

6.2. Verification

6.3. Clustering

.

6.4. Image retrieval

6.5. Per-class thresholds

.

β		Accuracy	AUC	100% - EER
Fixed	$\beta = 0.5$			
	$\beta = 1.0$			
	$\beta = 1.5$			
Learned	$\nu_\beta = 1.0$			
	$\nu_\beta = y$			
	$\nu_\beta = z$			

Table 3: Ablation study on LFW.

	# training images	Accuracy (%)
FaceNet [19]	200M	99.63
DeepFace [25]	4.4M	97.35
MultiBatch [24]	2.6M	98.20
VGG [18]	2.6M	99.13
WebFace [31]	494k	96.13
WebFace+PCA [31]	494k	96.30
WebFace+Joint Bayes [31]	494k	97.30
LightenedCNN [29]	494k	98.13
Npairs [21]	494k	98.33
Ours	494k	

Table 4: LFW. For the purpose of comparing algorithms, we only compare with results that were trained on the same dataset as our model. A few other state-of-the-art results were are listed for reference.

k	1	2	4	8	16	32
			Original Images			
Triplet Semihard [19, 22]	51.5	63.8	73.5	82.4	-	-
LiftedStruct [17, 22]	53.0	65.7	76.0	84.3	-	-
Npairs [21, 22]	53.9	66.8	77.8	86.4	-	-
StructClustering [22]	58.1	70.6	80.3	87.8	-	-
Ours	**65.5**	**76.4**	**85.0**	**90.8**	**94.7**	**97.3**
			Cropped Images			
PDDM Triple [11]	46.4	58.2	70.3	80.1	88.6	92.6
PDDM Quadruplet [11]	57.4	68.6	80.1	89.4	92.3	94.9
Ours	**73.1**	**82.5**	**89.0**	**93.6**	**96.6**	**98.5**

Table 5: CARS196

6.6. Convergence speed and ablation studies

.

References

[1] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98, 2015. 1

k	1	2	4	8	16	32
			Original Images			
Histogram [26]	52.8	64.4	74.7	83.9	90.4	94.3
Binomial Deviance [26]	50.3	61.9	72.6	82.4	88.8	93.7
Triplet Semihard [19, 22]	42.6	55.0	66.4	77.2	-	-
LiftedStruct [17, 22]	43.6	56.6	68.6	79.6	-	-
Npairs [21, 22]	45.4	58.4	69.5	79.5	-	-
StructClustering [22]	48.2	61.4	71.8	81.9	-	-
Ours	**54.7**	**67.3**	**77.9**	**86.4**	**92.6**	**96.0**
			Cropped Images			
PDDM Triplet [11]	50.9	62.1	73.2	82.5	91.1	94.4
PDDM Quadruplet [11]	58.3	69.2	79.0	88.4	93.1	95.7
Ours	**56.8**	69.1	**79.4**	87.0	92.3	**95.9**

Table 6: CUB200.

k	1	10	100	1000
Histogram [26]	63.9	81.7	92.2	97.7
Binomial Deviance [26]	65.5	82.3	92.3	97.6
Triplet Semihard [19, 22]	66.7	82.4	91.9	-
LiftedStruct [17, 22]	62.5	80.8	91.9	-
Npairs [21, 22]	66.4	83.2	93.0	-
StructClustering [22]	67.0	83.7	93.2	-
Ours	**71.0**	**85.1**	**93.4**	**97.9**

Table 7: Stanford Online Products.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009. 3

[3] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993. 1

[4] M. Bucher, S. Herbin, and F. Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *European Conference on Computer Vision*, pages 730–746. Springer, 2016. 1

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005. 1, 4

[6] P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994. 1

[7] R. Gao, D. Jayaraman, and K. Grauman. Object-centric representation learning from unlabeled videos. *arXiv preprint arXiv:1612.00500*, 2016. 2, 3

[8] M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. Where to buy it: Matching street clothing photos in

Figure 5: 5 classes with the smallest class-specific βs (left 5 columns), and 5 with the largest class-specific βs (right 5 columns). The former are clearly simpler in terms of colors, patterns, even activities, and backgrounds. TODO: add numbers. (CY: probably group them into 2 subfigs?)

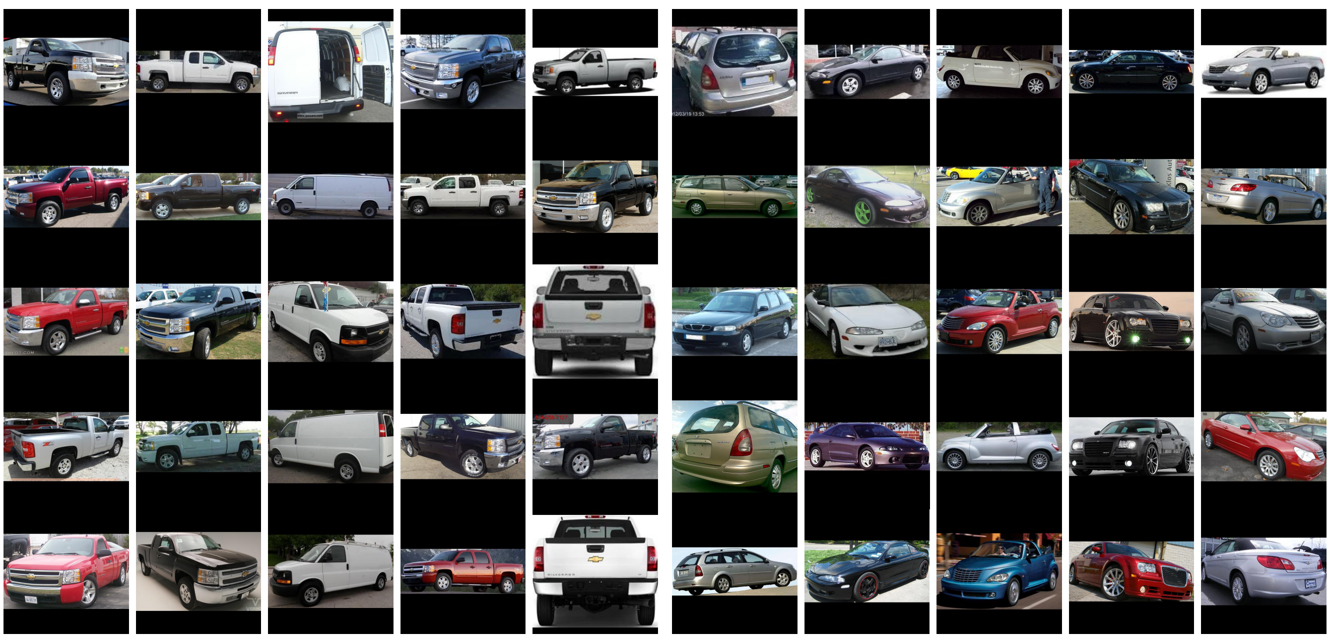


Figure 6: 5 classes with the smallest class-specific βs (left 5 columns), and 5 with the largest class-specific βs (right 5 columns). Note that for small-β cars are mostly *pickup trucks*, which have relatively simple colors and shapes, while the large-β cars are diverse in shapes and colors (even wheel colors). The first and the third largest are *convertibles*.

online shops. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3343–3351, 2015. 1

[9] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006. 1

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1

[11] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *Advances in Neural Information Processing Systems*, pages 1262–1270, 2016. 1, 5

[12] 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. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. 4

[13] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann. Self-paced curriculum learning. In *AAAI*, 2015. 3

[14] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 4

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[16] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010. 4

[17] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 1, 4, 5

[18] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015. 1, 2, 3, 5

[19] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 1, 2, 3, 5

[20] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, volume 1, page 2, 2003. 1, 2

[21] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1849–1857, 2016. 1, 3, 5

[22] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Learnable structured clustering framework for deep metric learning. *arXiv preprint arXiv:1612.01213*, 2016. 5

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1

[24] O. Tadmor, T. Rosenwein, S. Shalev-Shwartz, Y. Wexler, and A. Shashua. Learning a metric embedding for face recognition using the multibatch method. In *Advances In Neural Information Processing Systems*, pages 1388–1389, 2016. 5

[25] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014. 5

[26] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016. 5

[27] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009. 1

[28] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 4

[29] X. Wu, R. He, and Z. Sun. A lightened cnn for deep face representation. In *2015 IEEE Conference on IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015. 5

[30] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, 2002. 1

[31] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 4, 5

[32] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. *arXiv preprint arXiv:1611.05720*, 2016. 1