Softmax Dissection: Towards Understanding Intra- and Inter-Class Objective for Embedding Learning

Lanqing He,* Zhongdao Wang,* Yali Li, Shengjin Wang
Department of Electronic Engineering, Tsinghua University
{hlq17, wcd17}@mails.tsinghua.edu.cn, liyali13@mail.tsinghua.edu.cn, wgsgj@tsinghua.edu.cn

Abstract

The softmax loss and its variants are widely used as objectives for embedding learning applications like face recognition. However, the intra- and inter-class objectives in Softmax are entangled, therefore a well-optimized inter-class objective leads to relaxation on the intra-class objective, and vice versa. In this paper, we propose to dissect Softmax into independent intra- and inter-class objective (D-Softmax) with a clear understanding. It is straightforward to tune each part to the best state with D-Softmax as objective.Furthermore, we find the computation of the inter-class part is redundant and propose sampling-based variants of D-Softmax to reduce the computation cost. The face recognition experiments on regular-scale data show D-Softmax is favorably comparable to existing losses such as SphereFace and ArcFace. Experiments on massive-scale data show the fast variants significantly accelerates the training process (such as $64\times$) with only a minor sacrifice in performance, outperforming existing acceleration methods of Softmax in terms of both performance and efficiency.

1 introduction

Recent years have witnessed the prosperous development of deep learning and its applications. Among them, embedding learning (Liu et al. 2017; Wang et al. 2018; Deng et al. 2018; Schroff, Kalenichenko, and Philbin 2015) (or deep metric learning (Oh Song et al. 2016; Sohn 2016; Wang et al. 2017b)) is one of the most challenging problems that attracts wide attention, and corresponding research findings are supporting many applications like face recognition and person re-identification (Fan et al. 2019; Xiang et al. 2018).

The objective of embedding learning is to learn a mapping function $f(\cdot;\boldsymbol{\theta}) : \mathcal{X} \rightarrow \mathbb{R}^n$ so that in the embedding space $\mathbb{R}^n$ the distance between similar data is close while the distance between dissimilar data is far. The most straightforward choice is to formulate the embedding learning problem as a classification problem by employing the softmax loss as the objective. For instance, in face recognition, faces of different persons are considered as different classes and a large Softmax is used for learning the face embedding.

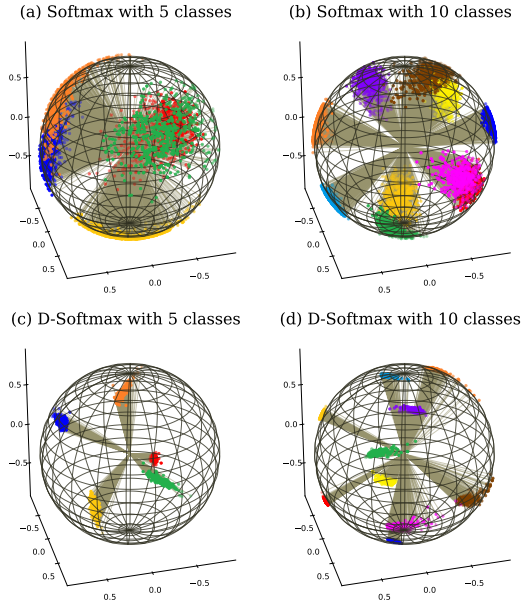

Figure 1: (a),(b): How intra- and inter-class objectives are entangled in Softmax. The inter-class distance in 5-class case is larger than that in 10-class case, therefore the constraint of intra-class objective is relaxed. (c),(d): The intra- and inter-class objectives are disentangled in D-Softmax. The intra-class distance is almost the same in both case.

However, there exist two major drawbacks in the softmax loss. First, the intra- and inter-class objectives are entangled. Such entanglement is visualized in Fig. 1 (a),(b). We select 5 and 10 identities respectively in the MS-Celeb-1M (Guo et al. 2016) dataset with the most samples, set the embedding dimension to 3 and plot the features. One can observe that with large inter-class distance the intra-class distance is also large. As we will show in Sec. 3, the reason is that the Softmax will gradually relax the intra-class objective along with the increase of inter-class distance, and vice versa. To our

*Equal contribution

knowledge, we are the first to discuss such entanglement, while existing works mostly address the insufficient discrimination issue by introducing additional supervision (Schroff, Kalenichenko, and Philbin 2015; Sun et al. 2014) or adding angular margin to the softmax loss (Liu et al. 2017; Wang et al. 2018; Deng et al. 2018).

Another shortage should be mentioned is time and memory cost. The softmax loss, as well as its variants, needs to compute class activations over *all* the classes. This leads to linear time and memory complexity w.r.t. the number of classes. In practice, the number of classes may be excessively large, say 10^6 or even beyond. The excessive memory demand makes it difficult to load all the class weights into the limited GPU memory, and the dramatically increased time cost is also not acceptable. Contrastive loss and triplet loss are possible alternatives that do not require much memory, but in terms of accuracy they significantly underperform the softmax family.

In this paper, we propose to dissect the softmax loss into intra- and inter-class objective. The intra-class objective pulls the feature close with the positive class-weight until a pre-defined criterion is satisfied, and the inter-class objective maintains the class weights to be widely separated in the embedding space. With the dissected softmax (D-Softmax) loss as the optimization objective, the intra- and inter-class objectives are disentangled, so that even the inter-class objective is well-optimized, the constraint on the intra-class objective is still rigorous (Fig. 1 (c),(d)).

Moreover, D-Softmax also dissects the computation complexity of Softmax into two independent parts. The intra-class objective only involves the sample and one positive class-weight, in contrast, the inter-class objective needs to compute activations over *all* negative classes. We find that such massive computation for the inter-class objective is somehow redundant. To facilitate the computation, we proposed to sample a *subset* of negative classes in one training pass. According to the difference in sampling strategies, we term the lightened D-Softmax as D-Softmax-B and D-Softmax-K respectively. Experiments show both strategies significant accelerate the training process with only a minor sacrifice in performance.

Our major contribution can be summarized as follows:

(1)We propose D-Softmax that dissects the intra- and inter-class objective of the softmax loss. The dissected intra-class objective is always rigorous, independent of how well the inter-class objective is optimized, and vice versa. Experiments show D-Softmax is favorably comparable with existing methods such as ArcFace on face recognition task.

(2)We make an important conclusion that the computation of inter-class objective is redundant and propose two sampling-based variants to facilitate the computation of D-Softmax. Training with massive classes (757K), our methods significantly accelerate the training process with only a minor sacrifice in performance.

2 Related Work

Softmax and its variants for face recognition. It is a widely adopted approach to formulate the face recognition as a multi-class classification problem. DeepFace (Taigman et al. 2014) and DeepID series (Sun, Wang, and Tang 2014; Sun et al. 2014; Sun, Wang, and Tang 2016) employ the conventional softmax loss in which the class activation is modeled as the inner product between vectors. Such loss is not effective enough, and some recent works address this problem by normalizing the embedding (Ranjan, Castillo, and Chellappa 2017) or the class-weights (Salimans and Kingma 2016). NormFace (Wang et al. 2017a) employs to normalize the both, which is equivalent to optimize the cosine distance. This inspired a series of works on softmax variants that optimize the angular distances between classes by introducing the angular margin (Liu et al. 2017; Wang et al. 2018; Deng et al. 2018). However, all aforementioned losses focus on strengthening the constraint but overlook a fact, that the insufficiency of Softmax is essentially caused by the entanglement of the intra- and inter-class objective.

Acceleration for Softmax. The acceleration of Softmax is an extensively studied problem typically in natural language processing, where large vocabularies need to be deal with. Existing methods mainly re-organize the structure of Softmax by the hierarchy of words (Goodman 2001), or the imbalanced frequency of classes (Schwenk 2007; Le et al. 2011; Chen, Grangier, and Auli 2015; Grave et al. 2017). However, these methods do not apply to real-world applications like face recognition, because the data are not hierarchical-structured nor substantially imbalanced on importance. HF-Softmax (Zhang et al. 2018) is a relatively related work to ours. It dynamically selects a subset of the training classes, by constructing a random forest in the embedding space and retrieving the approximate nearest neighbors. The time cost of loss computation is indeed reduced, but the update of the random forest still cost too much time. In this work, the light version D-Softmax do not require any extra computation besides the loss itself, so the computation is much faster. Moreover, the dissected intra-class objective is always rigorous, thus the performance is also superior.

3 Softmax Dissection

3.1 Preliminary Knowledge

The softmax Cross-Entropy loss is fomulated as,

$$\mathcal{L}_s = -\log(\frac{e^{sz_y}}{\sum_{i=1}^{K} e^{sz_k}}) = \log(1 + \frac{\sum_{k\neq y}^{K} e^{sz_k}}{e^{sz_y}}) \quad (1)$$

where s is a scale parameter, z_k indicates the activation of the k-th class, $k \in \{1, 2, ..., K\}$ and K is the number of classes. We denote the activation of the ground-truth class as z_y. In conventional Softmax, $z_k = \boldsymbol{w}_k^T \boldsymbol{x}$, where $\boldsymbol{w}_k$ is the class weight and $\boldsymbol{x}$ is the feature of the last fully connected layer. In recent arts *e.g*. NormFace (Wang et al. 2017a), the activation is usually modified as $z_k = \cos(\theta_{\boldsymbol{w}_k, \boldsymbol{x}})$. We adopt this cosine formulation for its good performance and intuitive geometric interpretation. Here we also list several variants of Softmax, *i.e*., SphereFace (Liu et al. 2017), ArcFace (Deng et al. 2018) and CosFace (Wang et al. 2018),

$$\mathcal{L}_{sphere} = \log(1 + \frac{\sum_{k\neq y}^{K} e^{sz_k}}{e^{s\cos(m_1 \arccos z_y)}}) \quad (2)$$

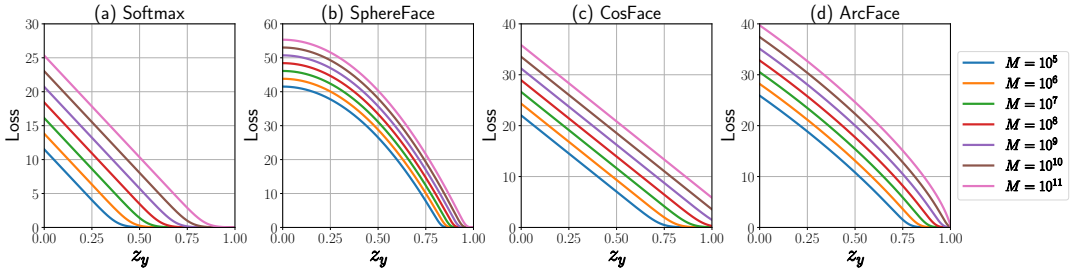

Figure 2: How the loss value varies with fixed inter-class similarity M against varying ground-truth class activation z_y in (a) Softmax, (b) SphereFace, (c) CosFace and (d) ArcFace. Different curves mean different M values.

$$\mathcal{L}_{arc} = \log(1 + \frac{\sum_{k \neq y}^{K} e^{sz_k}}{e^{s[\cos(\arccos z_y) + m_2]}}) \tag{3}$$

$$\mathcal{L}_{cos} = \log(1 + \frac{\sum_{k \neq y}^{K} e^{sz_k}}{e^{s(z_y - m_3)}}) \tag{4}$$

where m_1, m_2 and m_3 are hyperparameters that control the inter-class margin. Note that the only difference between these loss functions is the denominator in the fraction.

3.2 The Intra-Class Component

In this section, we first introduce how the intra-class objective is entangled with the inter-class objective. Then we compare the intra-class objective between Softmax and its margin-based variants (Eq.2-4). Finally, we present the intra-class objective of our Dissected Softmax loss.

Let $M = \sum_{k \neq y}^{K} e^{sz_k}$ represent the numerator in the fraction in the loss. M reflects the inter-class similarity. Large M means that the input has large cosine-similarity with all negative classes. With fixed M, we plot the loss $\mathcal{L}_s = \log(1 + \frac{M}{e^{sz_y}})$ against the ground-truth class activation z_y in Fig. 2 (a). Two observations can be made.

First, this family of curves can be approximated by piecewise linear functions: when z_y is small, $\mathcal{L}_s \to \log M - sz_y$, and when z_y is large, $\mathcal{L}_s \to 0$. It implies that when the intra-class similarity z_y is small, the loss will back-propagate a near-constant gradient, while the gradient is almost 0 when z_y is large. Second, the inflection point where the gradient changes is positively correlated to M.We can figure out the intersection point d of the piecewise linear function:

$$d = \frac{\log M}{s} \tag{5}$$

d can be considered as an approximate termination point of optimization because the gradient vanishes.

This observation supports an important conclusion:

- **Conclusion #1**: With Softmax as objective, when the class weights are widely separated (lead to small M), the optimization of intra-class objective almost terminates at a small value.

Unfortunately, the condition that *the class weights are widely separated*(explained in Sec. 3.4) always holds in the training process. Therefore, the termination of intra-class objective optimization is always so early that the training is not sufficient. By comparing the loss curves in Fig. 2, we speculate the early termination of the intra-class similarity optimization is the main reason why Softmax underperforms its margin-based variants. All termination points of variant curves have significant positive shifts compared to the vanilla Softmax under the same M. This means these losses do not stop optimizing the intra-class similarity until z_y is pretty large (say 0.8).

To address this problem that M is not large enough, we propose to disentangle the intra-class objective from the inter-class objective, by replacing M with a constant value ϵ. In this manner, we can manually adjust the optimization termination point d of the intra-class similarity to a sufficiently large value. To summarize, the intra-class component of the Dissected Softmax is:

$$\mathcal{L}_D^{intra} = \log(1 + \frac{\epsilon}{e^{sz_y}}) \tag{6}$$

3.3 The Inter-Class Component

In Sec. 3.2 we modified softmax loss and obtain a disentangled intra-class objective. However, we still need inter-class objective as a *regularization* to avoid collapsing to a trivial solution where all the data is mapped to a single point. Similarly, we first analyze the inter-class objective of softmax and its variants, then give the formulation of the inter-class objective of D-Softmax.

Consider a sample x of class y and its activation on the n-th ($n \neq y$) class z_n. Softmax loss can be written as,

$$\mathcal{L}_s = \log(1 + \frac{e^{sz_n} + \sum_{k \neq y,n}^{K} e^{sz_k}}{e^{sz_y}}) = \log(1 + \frac{e^{sz_n} + M_n}{e^{sz_y}}) \tag{7}$$

where we replace the summation with M_n for convenience.

Firstly we fix M_n and study how the loss varies with different z_n and z_y. A family of curves are presented in Fig. 3 (a). Similar characteristic emerges like in the intra-class analysis: The gradient $\frac{\partial \mathcal{L}_s}{\partial z_n}$ remains almost constant with large negative-class similarity z_n and diminishes rapidly to 0 at some point. Once again we define the optimization termination point for z_n as the intersection point of the approximate piecewise linear function,

$$d' = \frac{\log(e^{sz_y} + M_n)}{s} \tag{8}$$

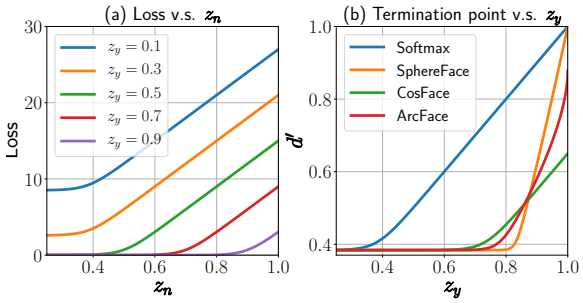

Figure 3: How the inter-class objective of Softmax is entangled with the intra-class objective. (a) The loss curve of Softmax against negative-class similarity z_n under different positive-class similarity z_y. (b) The termination point of the inter-class objective w.r.t. varying intra-class similarity z_y.

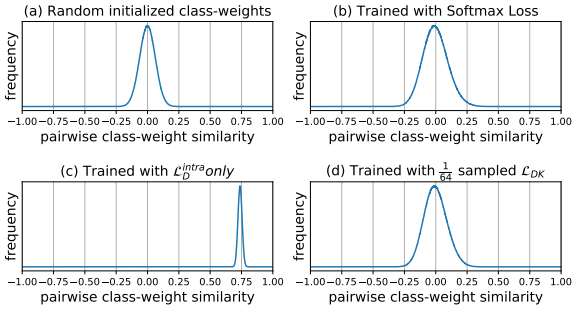

Figure 4: The distributions of pairwise class-weight cosine similarities (a) when the class-weights are randomly initialized, (b) after trained with Softmax, (c) after trained with the intra-class objective $\mathcal{L}_D^{intra}$ only, (d) after trained with the $\frac{1}{64}$ sampled $\mathcal{L}_D^{inter}$.

and a conclusion can be drawn,

- **Conclusion #2**: With Softmax as objective, when the intra-class similarity (z_y) is large, the optimization of the negative class weights almost terminates at a large value.

This may lead to non-sufficient discrepancy among different class weights thus hamper the embedding learning. As an evidence, we plot in Fig. 3 (b) the termination point d' against the intra-class similarity z_y for Softmax, SphereFace, CosFace and ArcFace. Wider plateau in the curve means the objective regularizes the inter-class similarity more rigorously. All the large-margin Softmax variants present much wider plateau than the vanilla Softmax.

In light of above analysis, we propose to disentangle the inter-class objective, by replacing the intra-class similarity e^{sz_y} with a constant. We simply set this constant to 1, therefore the inter-class component of the Dissected Softmax is,

$$\mathcal{L}_D^{inter} = \log(1 + \sum_{k \neq y} e^{sz_k}) \tag{9}$$

In such manner, the d' curve is a flat line, which means the regularization on inter-class similarity is always strict.

3.4 D-Softmax and Its Light Variants

Based on Eq.6 and 9, the final form of the Dissected Softmax (D-Softmax) loss is,

$$\mathcal{L}_D = \mathcal{L}_D^{intra} + \mathcal{L}_D^{inter} = \log(1 + \frac{\epsilon}{e^{sz_y}}) + \log(1 + \sum_{k \neq y} e^{sz_k}) \tag{10}$$

The merits of Dissected Softmax are mainly two-folds. First, as we learn from Conclusion #1 and #2 that, in vanilla Softmax, the optimization of intra- and inter-class objective is entangled. Minimizing the intra-class objective will relax the regularization on inter-class objective, and vice versa. In D-Softmax, the optimization is disentangled, thus the constraints are always strict, so the learned embedding is more discriminative. Second, such disentangled formulation allows us to further reduce the computational complexity of the loss function, significantly boosting the training efficiency when the number of classes is tremendous.

When the number of classes is larger than 10^6, the computation of Softmax becomes the bottleneck of the training process. Denote the batch size as B, the number of classes as K, then the time complexity for computing Softmax is $\mathcal{O}(BK)$. In D-Softmax, this complexity is dissected into $\mathcal{O}(B)$ for $\mathcal{L}_D^{intra}$ plus $\mathcal{O}(B(K-1))$ for $\mathcal{L}_D^{inter}$. When $K \gg B$, the computation of $\mathcal{L}_D^{inter}$ becomes the major time overhead. In order to accelerate training, let us consider: Is it necessary to compute all the negative-class activations in a mini-batch?

In this work, our answer is **No**. The main reason lies in the approximate orthogonality of class-weights in high-dimensional space. For illustrating the approximate orthogonality, we randomly initialize $10,000$ class weights with 256 dimension and plot how the pairwise cosine similarities distribute in Fig. 4 (a). The pairwise cosine similarities present a narrow Gaussian distribution with zero mean and around $3\sigma = 0.2$, which means the class weights are far apart from each other. We also plot how this distribution changes after training with softmax in Fig. 4 (b). Interestingly, the mean of the Gaussian distribution does not shift, and the variance just increases a little. This means $\mathcal{L}_D^{inter}$ is not pushing the class weights strictly further from each other. Considering above two points, we may reach the following conclusion,

- **Conclusion #3**: When optimizing in high-dimensional embedding space, the function of the inter-class objective is not pushing class-weights strictly further apart, but mainly maintaining the approximate orthogonality of the class-weights as a regularization.

Based on this conclusion, we speculate the $\mathcal{O}(B(K-1))$ computation of $\mathcal{L}_D^{inter}$ may be redundant. To validate it, we again train an identical model using $\mathcal{L}_D^{intra}$ and a sampled $\mathcal{L}_D^{inter}$. In each mini-batch we randomly sample $\frac{1}{64}$ of the $K-1$ classes as the negative classes. After training , we plot the distribution of pairwise cosine similarities between class weights in Fig. 4 (d). As expected, the distribution is almost the same as training with the full Softmax. In Sec. 4 we will present the performance degradation of the sampled loss compared to the full D-Softmax is also minor while the computation of $\mathcal{L}_D^{inter}$ is $64\times$ faster. We name this light variant of D-Softmax as D-Softmax-K for the negative classes are

Table 1: Face recognition performance with different loss functions and performance of D-Softmax with different configurations. The best results are **bolded** and the second best results are underlined.

Loss	d	Verification accuracy (%)			IJBC:TAR@FAR (%)				MegaFace:
		LFW	CFP	AgeDB	10^{-1}	10^{-2}	10^{-3}	10^{-4}	Rank1@10^6
$\mathcal{L}_s$	-	99.30	87.23	94.48	**98.15**	95.82	91.45	85.43	91.25
$\mathcal{L}_{sphere}$	-	99.59	91.37	96.62	98.03	96.10	92.60	86.41	96.04
$\mathcal{L}_{arc}$	-	99.68	92.26	97.23	98.01	95.84	92.64	87.29	96.97
$\mathcal{L}_D$	0.5	99.38	88.34	95.04	98.02	95.80	91.45	85.47	92.06
$\mathcal{L}_D$	0.7	99.60	91.44	96.51	98.08	96.15	92.58	86.56	95.90
$\mathcal{L}_D$	0.9	**99.74**	92.27	97.22	98.09	**96.21**	92.91	88.17	96.94
$\mathcal{L}_D$	1.0	99.63	92.01	96.88	98.03	96.11	92.60	86.88	96.25
$\mathcal{L}_D^{intra} + \mathcal{L}_s^{inter}$	0.9	99.47	90.21	95.21	98.01	95.94	91.78	85.86	93.08
$\mathcal{L}_D^{intra} + \mathcal{L}_{arc}^{inter}$	0.9	99.73	**93.07**	**97.30**	98.02	96.12	**92.97**	**88.28**	**97.02**

sampled from the $K-1$ classes. Formally, The mini-batch version of D-Softmax-K is

$$\mathcal{L}_{DK} = \sum_{i=1}^{B} \log(1 + \frac{\epsilon}{e^{sz_{y_i}}}) + \sum_{i=1}^{B} \log(1 + \sum_{k \in \mathcal{S}_K} e^{sz_k}) \quad (11)$$

where $\mathcal{S}_K = \{k|k = 1, 2, ..., K\} \backslash \{y_i|i = 1, 2, ..., B\}$ means a subset of the class-weight set . The sampling rate remains a hyperparameter for performance-speed trade-off.

An alternative sampling strategy is sampling from mini-batch samples, and we name such strategy as D-Softmax-B,

$$\mathcal{L}_{DB} = \sum_{i=1}^{B} \log(1 + \frac{\epsilon}{e^{sz_{y_i}}}) + \sum_{i \in \mathcal{S}_B} \log(1 + \sum_{k=1}^{K} e^{sz_k}) \quad (12)$$

where $\mathcal{S}_B$ is a subset of batch samples. The strengths and weaknesses of each strategy will be shown in Sec. 4.3.

4 Experimental Results

4.1 Datasets and Evaluation Metrics

Evaluation. We validate the effectiveness of the proposed D-Softmax in the face recognition task. The testing datasets include LFW (Huang et al. 2008), CFP-FP (Sengupta et al. 2016), AgeDB-30 (Moschoglou et al. 2017), IJB-C (Maze et al. 2018) and MegaFace (Kemelmacher-Shlizerman et al. 2016). LFW is a standard face verification benchmark that includes 6,000 pairs of faces, and the evaluation metric is the verification accuracy via 10-fold cross validation. CFP-FP and AgeDB-30 are similar to LFW but emphasis on frontal-profile and cross-age face verification respectively. IJB-C is a large-scale benchmark for template-based face recognition. A face template is composed of multiple face images or video face tracks. Features are simply average pooled in a template to obtain the template feature. The evaluation metric is the true accept rate (TAR) at different false alarm rate (FAR). MegaFace identification challenge is a large-scale benchmark to evaluate the performance at the million distractors. We perform the rank-1 identification accuracy with 10^6 distractors on the a refined version used by ArcFace[1].

[1]https://github.com/deepinsight/insightface

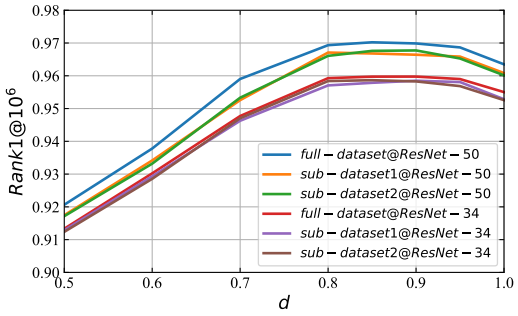

Figure 5: Rank-1 identification accuracy against 10^6 distractors on MegaFace dataset (refined) with different hyperparameter.We randomly split the full MS-Celeb-1M dataset into sub-dataset1 and sub-dataset2

Training. We adopt the MS-Celeb-1M (Guo et al. 2016) dataset for training. Since the original MS-Celeb-1M contains wrong annotations, we adopt a cleaned version that is also used in ArcFace. The cleaned MS-Celeb-1M consists of around 5.8M images of 85K identities. Moreover, to validate the effectiveness and efficiency of the proposed losses on massive-scale data, we combine MS-Celeb-1M with the MegaFace2 (Nech and Kemelmacher-Shlizerman 2017) dataset to obtain a large training set. The MegaFace2 dataset consists of 4.7M images of 672K identities, so the joint dataset has 9.5M images of 757K identities in total.

4.2 Experiments on D-Softmax

In this section, we explore how to set the intra-class termination point d for best performance, and how different formulations of inter-class objective affect the discrimination of the learned embedding. Finally we compare D-Softmax with other state-of-the-art loss functions.

All the models are standard ResNet-50 (He et al. 2016), trained on MS-Celeb-1M. We set the scale $s = 32$, the margin $m_1 = 4$ for SphereFace, $m_2 = 0.5$ for ArcFace for the best performance. The other hyperparameters are the same.

Selection of d. By tuning the hyperparameter ϵ in $\mathcal{L}_D$, we are able to set the optimal d. Table 1 shows performance of $\mathcal{L}_D$ with different settings of d. With d increasing from 0.5

Table 2: Comparison of D-Softmax-B, D-Softmax-K and other sampling-based Softmax variants in terms of face recognition accuracy. The best results at $1/64$ sampling rate are **bolded**, and the second best results are underlined.

Loss	$\mid \mathcal{S}_B \mid$	$\mid \mathcal{S}_k \mid$	Sampling Rate	Verification accuracy (%)			IJB:TAR@FAR (%)				MegaFace: Rank1@10^6
				LFW	CFP	AgeDB	10^{-1}	10^{-2}	10^{-3}	10^{-4}	
D-Softmax-B	256	85K	1	99.74	92.27	97.22	98.09	96.21	92.91	88.17	96.94
D-Softmax-B	64	85K	1/4	99.75	92.27	97.18	98.08	96.22	92.90	88.13	96.93
D-Softmax-B	16	85K	1/16	99.74	92.24	96.92	98.03	96.20	93.02	87.98	96.91
D-Softmax-B	4	85K	1/64	**99.60**	**90.89**	**95.84**	98.09	**95.87**	**92.15**	**86.74**	**95.34**
D-Softmax-B	1	85K	1/256	99.50	89.09	94.57	97.95	95.29	91.25	85.24	91.53
D-Softmax-K	256	1.3K	1/64	99.55	89.77	95.02	98.09	95.40	92.01	86.03	94.72
Rand-Softmax	256	1.3K	1/64	99.07	85.47	89.35	98.05	94.30	87.52	78.96	88.27
Rand-ArcFace	256	1.3K	1/64	99.43	88.21	84.08	**98.11**	95.14	91.27	84.26	93.62
HF-Softmax	256	1.3K	1/64	99.18	86.11	91.55	97.92	94.45	89.63	81.85	91.18

to 0.9, the performance increases steadily. However, when we further insrease d to 1.0, the performance drops slightly. We also perform a range of experiences with different backbone networks and training set and Fig.5 shows how the identification accuracy varies with different d. In each setting, the conclusion is consistent: A moderately large termination point for intra-class similarity *e.g.* 0.9, leads to the best results, so we set $d = 0.9$ in all the following experiments.

Different forms of inter-class objective. Apart from the simple form $\mathcal{L}_D^{inter}$ proposed in Sec. 3.3, we also compare several different forms of inter-class objective, the inter-class objective of Softmax(NormFace) and ArcFace. We donate such objectives as $\mathcal{L}_s^{inter}$ and $\mathcal{L}_{arc}^{inter}$.

To accomplish such objectives, in the forward pass we compute the full Softmax or ArcFace loss, while in the backward pass we only back-propagate the inter-class part of the gradients, by setting $\frac{\partial \mathcal{L}}{\partial z_y}$ to 0. Then we combine $\mathcal{L}_s^{inter}$ or $\mathcal{L}_{arc}^{inter}$ with the intra-class part of D-Softmax $\mathcal{L}_D^{intra}$ to train a model. Table 1 compares the performance between $\mathcal{L}_D$, $\mathcal{L}_D^{intra} + \mathcal{L}_s^{inter}$ and $\mathcal{L}_D^{intra} + \mathcal{L}_{arc}^{inter}$. With the same intra-class objective, it is shown that $\mathcal{L}_D^{inter}$ outperforms $\mathcal{L}_s^{inter}$ by a large margin. $\mathcal{L}_D^{inter}$ and $\mathcal{L}_{arc}^{inter}$ lead to almost the same good performance, which is as expected. Though entangled with the intra-class objective, the regularization on inter-class similarity in ArcFace is rigorous enough until the intra-class similarity is pretty large (say > 0.8). Similarly, the proposed dissected form of inter-class objective is always rigorous regardless of the intra-class similarity. Compared with $\mathcal{L}_{arc}^{inter}$, $\mathcal{L}_D^{inter}$ has a more concise form with no hyperparameter and is easier to be extended to fast variants.

Comparison with state-of-the-art losses. For fair comparison, we re-implement NormFace (Wang et al. 2017a), SphereFace and ArcFace and compare the proposed D-Softmax with them using the same training data and model. As shown in Table 1, the proposed D-Softmax outperforms the Softmax (NormFace) baseline even with a small $d = 0.5$, and with $d = 0.9$ D-Softmax outperforms Softmax by a significant margin. SphereFace and ArcFace also outperform the Softmax baseline because of the introduced angular margin. To tell the difference between d and margin parameteter, we take ArcFace for example. The hyperparameter m is the required angular margin between sample features and negative class weights, affects both the intra- and inter-class objective, so that for all class the intra-class constraint is not the same rigorous. Therefore, it needs to be tuned. The best selection may vary with other hyperparameters varies. Instead, we introduce d which has a more clear interpretation to reach the same goal of adding margin. d indicates the optimization termination of distance between sample features and positive class weights. Therefore, it is straightforward to select a reasonable value of d, and for all class the intra-class constraint is the same rigorous. The best selection of d in D-Softmax consistently ranges from 0.8 to 0.9 with different training settings, even when we down-sample the inter-class objective in the next section.

4.3 Experiments on Light D-Softmax

In Sec. 3.4 we proposed two sampling-based variants of D-Softmax, *i.e.*, D-Softmax-B and D-Softmax-K, for reducing the computational complexity of training with massive classes. In this section, we explore the strength and weakness of each sampling strategy.

D-Softmax-B. D-Softmax-B is a most easy-to-implement sampling method for reducing the complexity of the inter-class objective. In practice, one only needs to sample from the batch samples and then computes all the negative-class activations w.r.t. the sampled batch samples. To illustrate the effectiveness of D-Softmax-B, we train several ResNet-50 with batch size of 256, and employ D-Softmax-B as the objective, with sampling rates varying from 1 to $1/256$. As shown in Table 2, the performance drops slowly until the sampling rate is lower than $1/64$. The accuracy drop of $1/16$ sampling rate is nearly neglectable compared to the non-sampled version, while the computation of $\mathcal{L}_D^{inter}$ is $16\times$ faster. However, even with the extreme sampling rate $1/256$, *i.e.*, only one batch sample is used for computing the inter-class objective, the performance of D-Softmax-B is still acceptable(99.50%v.s. full-computed version 99.74% LFW accuracy). These results in turn strongly support Conclusion #3 we made in Sec. 3.4, that the inter-class objective is mainly maintaining the approximate orthogonality of class weights as a regularization, thus the full-computation with $\mathcal{O}(B(K-1))$ is redundant. The advantages of D-Softmax-B are the simplicity for implementation and minimal sacrifice of performance. However, it faces a dilemma in practice, *i.e.*, the memory limit of GPU is also a matter in large-scale training. The computation of D-Softmax-B requires the whole

Table 3: Comparison between D-Softmax-K and several baseline methods on large-scale training set. The loss/total average time is computed as the average time for one forward-backward pass of the loss layer / the entire model. The best results are **bolded** and the second best results are underlined.

Loss	Loss Avg. Time (s)	Total Avg. Time (s)	Verification accuracy (%)			IJBC:TAR@FAR (%)				MegaFace: Rank1@10^6
			LFW	CFP	AgeDB	10^{-1}	10^{-2}	10^{-3}	10^{-4}	
Softmax	3.12	3.96	99.38	87.96	**95.60**	**98.14**	**95.84**	91.55	85.79	93.03
Rand-Softmax	**0.20**	**1.04**	99.10	85.56	89.58	98.06	94.44	88.02	79.21	88.92
HF-Softmax	2.04	2.88	99.27	86.10	91.82	98.04	94.71	90.26	82.23	91.79
D-Softmax-K	0.21	1.05	**99.47**	**89.59**	95.32	98.10	95.66	**91.83**	**85.83**	**94.54**

class-weight matrix to be copied to the GPU memory thus adds difficulties on parallelism.

D-Softmax-K. For each mini-batch, D-Softmax-K first samples candidate negative classes from the intersection of negative-classes sets w.r.t. every batch sample, then the batch inter-class objective is computed with simple data parallel. To tackle the problem of GPU memory limit, inspired by (Zhang et al. 2018), we adopt a parameter server to store all the class weights on a large-capacity memory (*e.g.* CPU Ram). When some classes are sampled in a mini-batch, the weights of these classes are retrieved on the parameter server and then cached in the client's GPU. In such manner the dilemma of GPU memory limit is mitigated, and also the implementation is not so complicated.

However, compared with D-Softmax-B at the same sampling rate (see the gray rows in Table 2), performance of D-Softmax-K is slightly inferior. A possible interpretation is that in D-Softmax-B all the class weights are updated in every mini-batch thus the class weights are more up-to-date in each iteration. This suggests sampling from the batch samples can achieve better performance. Nevertheless, considering the difference in performance is minor while D-Softmax-K is much easier for parallelism, we suggest to use D-Softmax-K in large-scale training.

Compared with other sampling-based methods. In order to demonstrate the benefits of D-Softmax, we also compare with some exsiting sampling-based methods. The first is random Softmax, which means for one mini-batch the to-be-computed class weights are randomly sampled. The second is random ArcFace, which is similar to Rand-Softmax but the loss function is ArcFace. At the same $1/64$ sampling rate, both D-Softmax-B and D-Softmax-K outperform Rand-Softmax and Rand-ArcFace by a significant margin ($99.60/99.55\%$ v.s. $99.07/99.43\%$ LFW accuracy). The sampling operation reduce the inter-class objective, which relaxes the intra-class constraint of Softmax and ArcFace. As for D-Softmax, the intra-class constraint is not affected.

HF-Softmax proposed in (Zhang et al. 2018) also needs to be comapred, so we adopt the code released by the authors and train HF-Softmax on the same dataset for fair comparison. It also samples from $K-1$ negative classes to reduce the computational cost. The difference is that the sampling is not random, they build a hash forest to partition the weight space and find approximate-nearest-neighbor (ANN) class weights for batch samples. As shown in Table 2, HF-Softmax outperforms Rand-Softmax (99.18% v.s. 99.07% LFW accuracy), since the negative class weights are sampled from the 'hard-negatives' which are more valuable for optimization. But compared with D-Softmax, the performance is inferior. It is again because the entanglement between the intra- and inter-class objective. Though hard negative class weights are mined, only the inter-class regularization is improved. The intra-class constraint is still not strict enough.

Large-scale experiments. In order to validate the acceleration effectiveness of the proposed D-Softmax-K on training, we perform a large-scale experiment on the joint dataset of MS-Celeb-1M and MegaFace2. Performance and average time cost of some baseline methods are listed in Table 3. The sampling rate is set to $1/64$ in all losses. HF-Softmax and D-Softmax outperform Rand-Softmax at the same sampling rate in terms of accuracy, yet only D-Softmax outperforms the full Softmax loss. Sampling based on the entangled form of Softmax, the performance upper bound of HF-Softmax is comaprable to the full Softmax. In contrast, the sampled D-Softmax has the ability to exceed full Softmax.

In terms of the time cost, it is obvious that the full Softmax is the slowest one, with 3.12s average time cost on the loss layer for one forward-backward pass, while Rand-Softmax is the fastest with 0.20s. HF-Softmax is supposed to be efficient because only a small fraction of the weights need to be computed, but the update of the random forest cost too much time (1.83s on average, while the computation of loss is only 0.21s.). This time cost can be decreased by changing to fast ANN algorithm or enlarging the updating time duration of the random forest, but the performance will decrease meanwhile. In contrast, the proposed D-Softmax-K provides a pretty good performance-speed trade-off. The training with D-Softmax-K is as fast as Rand-Softmax since we do not need to build and update a random forest.

Note that the results of large-scale experiments seem to be inferior to that of training with MS-Celeb-1M alone. This is because the MegaFace2 dataset is rather noisy. If trained with a cleaned large-scale dataset, the performance is supposed to be better.

5 Conclusion

In this paper, we propose to dissect the softmax loss into independent intra- and inter-class objectives. By doing so, the optimization of the two objectives is no longer entangled with each other, and as a consequence it is more straightforward to tune the objectives to be consistently rigorous during the training time. The propsed D-Softmax shows good performance in the face recognition task. By sampling the inter-class similarity, it is easy to be extended to fast variants

(D-Softmax-B and D-Softmax-K) that can handle massive-scale training. We show that the fast variants of D-Softmax significantly accelerate the training process, while the performance drop is quite small.

References

Chen, W.; Grangier, D.; and Auli, M. 2015. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*.

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2018. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*.

Fan, X.; Jiang, W.; Luo, H.; and Fei, M. 2019. Spherereid: Deep hypersphere manifold embedding for person re-identification. *Journal of Visual Communication and Image Representation*.

Goodman, J. 2001. Classes for fast maximum entropy training. *arXiv preprint cs/0108006*.

Grave, E.; Joulin, A.; Cissé, M.; Jégou, H.; et al. 2017. Efficient softmax approximation for gpus. In *ICML*.

Guo, Y.; Zhang, L.; Hu, Y.; He, X.; and Gao, J. 2016. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Huang, G. B.; Mattar, M.; Berg, T.; and Learned-Miller, E. 2008. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*.

Kemelmacher-Shlizerman, I.; Seitz, S. M.; Miller, D.; and Brossard, E. 2016. The megaface benchmark: 1 million faces for recognition at scale. In *CVPR*.

Le, H.-S.; Oparin, I.; Allauzen, A.; Gauvain, J.-L.; and Yvon, F. 2011. Structured output layer neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; and Song, L. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*.

Maze, B.; Adams, J.; Duncan, J. A.; Kalka, N.; Miller, T.; Otto, C.; Jain, A. K.; Niggel, W. T.; Anderson, J.; Cheney, J.; et al. 2018. Iarpa janus benchmark-c: Face dataset and protocol. In *2018 International Conference on Biometrics (ICB)*.

Moschoglou, S.; Papaioannou, A.; Sagonas, C.; Deng, J.; Kotsia, I.; and Zafeiriou, S. 2017. Agedb: the first manually collected, in-the-wild age database. In *CVPR Workshops*.

Nech, A., and Kemelmacher-Shlizerman, I. 2017. Level playing field for million scale face recognition. In *CVPR*.

Oh Song, H.; Xiang, Y.; Jegelka, S.; and Savarese, S. 2016. Deep metric learning via lifted structured feature embedding. In *CVPR*.

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

Salimans, T., and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*.

Schwenk, H. 2007. Continuous space language models. *Computer Speech & Language* 21(3):492–518.

Sengupta, S.; Chen, J.-C.; Castillo, C.; Patel, V. M.; Chellappa, R.; and Jacobs, D. W. 2016. Frontal to profile face verification in the wild. In *WACV*.

Sohn, K. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*.

Sun, Y.; Chen, Y.; Wang, X.; and Tang, X. 2014. Deep learning face representation by joint identification-verification. In *NIPS*.

Sun, Y.; Wang, X.; and Tang, X. 2014. Deep learning face representation from predicting 10,000 classes. In *CVPR*.

Sun, Y.; Wang, X.; and Tang, X. 2016. Sparsifying neural network connections for face recognition. In *CVPR*.

Taigman, Y.; Yang, M.; Ranzato, M.; and Wolf, L. 2014. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*.

Wang, F.; Xiang, X.; Cheng, J.; and Yuille, A. L. 2017a. Normface: l 2 hypersphere embedding for face verification. In *ACM MultiMedia*.

Wang, J.; Zhou, F.; Wen, S.; Liu, X.; and Lin, Y. 2017b. Deep metric learning with angular loss. In *ICCV*.

Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; and Liu, W. 2018. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*.

Xiang, W.; Huang, J.; Qi, X.; Hua, X.; and Zhang, L. 2018. Homocentric hypersphere feature embedding for person re-identification. *arXiv preprint arXiv:1804.08866*.

Zhang, X.; Yang, L.; Yan, J.; and Lin, D. 2018. Accelerated training for massive classification via dynamic class selection. In *AAAI*.