Virtual Class Enhanced Discriminative Embedding Learning

Binghui Chen¹, Weihong Deng¹, Haifeng Shen²
¹Beijing University of Posts and Telecommunications
²AI Labs, Didi Chuxing, Beijing 100193, China

chenbinghui@bupt.edu.cn, whdeng@bupt.edu.cn, shenhaifeng@didiglobal.com

Abstract

Recently, learning discriminative features to improve the recognition performances gradually becomes the primary goal of deep learning, and numerous remarkable works have emerged. In this paper, we propose a novel yet extremely simple method Virtual Softmax to enhance the discriminative property of learned features by injecting a dynamic virtual negative class into the original softmax. Injecting virtual class aims to enlarge inter-class margin and compress intra-class distribution by strengthening the decision boundary constraint. Although it seems weird to optimize with this additional virtual class, we show that our method derives from an intuitive and clear motivation, and it indeed encourages the features to be more compact and separable. This paper empirically and experimentally demonstrates the superiority of Virtual Softmax, improving the performances on a variety of object classification and face verification tasks.

1 Introduction

In the community of deep learning, the Softmax layer is widely adopted as a supervisor at the top of the model due to its simplicity and differentiability, such as in object classification⁹¹¹³⁹ and face recognition²⁹³⁰²⁷³⁶, etc. While, many research works take it for granted and omit the fact that the learned features by Softmax are only separable not discriminative. Thus, the performances of deep models in many recognition tasks are limited. Moreover, there are a few research works concentrating on learning discriminative features through refining this commonly used softmax layer, e.g. L-Softmax⁴² and A-Softmax²¹. However, they require an annealing-like training procedure which is controlled by human, and thus are difficult to transfer to other new tasks. To this end, we intend to propose an automatic counterpart which dedicates to learning discriminative features.

In standard softmax, the input pattern \( x_i \) (with label \( y_i \)) is classified by evaluating the inner product between its feature vector \( X_i \) and the class anchor vector \( W_j \) (if not specified, the basis \( b \) is removed, which does not affect the final performances of DCNN verified in ²²²¹). Since the inner product \( W_j^T X_i \) can be rewritten into \( \| W_j \| \| X_i \| \cos \theta_j \), where \( \theta_j \) is the angle between vector \( X_i \) and \( W_j \), and the linear softmax classifier is mainly determined by the angle, thus can be regarded as an angle classifier. Moreover, as shown in ³³, the feature distribution leaned by softmax is 'radial', like in Fig.1. Thus, learning the angularly discriminative feature is the bedrock of softmax-based classifier. Here, we will first succinctly elucidate the direct motivation of our Virtual Softmax. For a 4-class classification problem, as illustrated in Fig.1(a), one can observe that when optimizing with the original 4 classes, the

\[^{1}\]Elucidated in ²². Here for simplicity, assume that all \( \| W_j \| = l \).

\[^{2}\]N-dimensionality space complicates our analysis but has the similar mechanism as 2-D space.

Figure 1: Illustration of angularly distributed features on 2-D space. (a) shows features learned by the original 4-way softmax. (b) shows features learned by 8-way softmax, where there are 4 additionally hand-injected negative classes. \( W \) denotes the class anchor vector.

decision boundary for class 1 overlaps with that for class 2, i.e. there is no inter-class angular margin. Obviously, this constraint of overlapped boundaries is not enough to ensure intra-class compactness and inter-class separability, and then contributes little to recognition performance improvement. While, in expanded 8-category constraint case as exhibited in Fig[1](b), the decision boundaries for class 1 and 2 have a large inter-class angular margin with each other due to the additionally injected class 5 (which doesn’t take part in the final recognition evaluation) and the constrained region of class 1 is much tighter than before (other classes have the same phenomenon), yielding more compact and separable features which are highly beneficial to the original 4-class recognition. Thus, artificially injected virtual classes (e.g. $W_5 \sim W_8$) can be dedicated to encouraging larger decision margin among original classes (e.g. class 1 $\sim$ 4) such that produces angularly discriminative features.

Naturally motivated by this, we propose Virtual Softmax, a novel yet extremely simple technique to improve the recognition performances by training with an extra dynamic virtual class. Injecting this virtual class aims at imposing stronger and continuous constraint on decision boundaries so as to further produces the angularly discriminative features. Concretely, generalizing Softmax to Virtual Softmax gives chances of enlarging inter-class angular margin and tightening intra-class distribution, and allows the performance improvement of recognition via introducing a large angular margin. Different from L-Softmax[22] and A-Softmax[21], our work aims at extending Softmax to an automatic counterpart, which can be easily performed without or with little human interaction. Fig[2] verifies our technique where the learned features by our Virtual Softmax turn to be much more compact and well separated. And the main contributions of this paper are summarized as follows:

- We propose Virtual Softmax to automatically enhance the representation power of learned features by employing a virtual negative class. The learned features are much more compact and separable, illustrated in Fig[2]. And to our best knowledge, it is the first work to employ additional virtual class in Softmax to optimize the feature learning and to improve the recognition performance.
- The injected virtual class derives from the natural and intuitive motivation, and intends to force a zero limit $\theta_{pj}$ which is a much stronger constraint than softmax. Moreover, it is dynamic and adaptive, pursuing an automatic training procedure and without incurring much computational complexity and memory consumption.
- Extensive experiments have been conducted on several datasets, including MNIST[17], SVHN[23], CIFAR10/100[16], CUB200[35], ImageNet32[5], LFW[12] and SLLFW[6]. Finally, our Virtual Softmax has achieved competitive and appealing performances, validating its effectiveness.

2 Related Work

Since the training of deep model is dominated and guided by the top objective loss, and this objective function can be equipped with more semantic information, many research works force stronger representation by refining the supervisor at the top of deep model. In [29], contrastive loss is employed to enlarge the inter-class Euclidean distance and to reduce the intra-class Euclidean distance. Afterwards, it is generalized to triplet loss, the major idea is to simultaneously maximize inter-class distance and minimize intra-class distance, and it is widely applied in numerous computer vision tasks, such as in face verification [27], in person re-identification [4], in fine-grained classification [34], etc. Moreover, many tuple-based methods are developed and perform better on the corresponding tasks, such as Lifted loss [24], N-pair loss [28][2], etc. However, these tuple-based methods constrain the feature learning with multiple instances, and thus require to elaborately manipulate the tuple mining procedure, which is much expensive in computation and is performance-sensitive. In addition, inspired by linear discriminant analysis, center loss [36] breaks away from the tuple-based idea and shows better performances. However, our Virtual Softmax differs with them in that (1) the model is optimized with only the single Virtual Softmax loss not the joint loss functions (e.g. softmax + contrastive-loss). (2) our method can also be applied to improve the classification performances, while the above methods are not as good in classification tasks. Compared to L-Softmax[22] and A-Softmax[21], this paper heads from a totally different idea that encourages a large margin among classes via injecting additionally virtual class, and dedicates to proposing an automatic method, i.e. Virtual-Softmax.

3 Intuition and Motivation

In this section, we will give a toy example to introduce the immediate motivation of our Virtual Softmax. Define the i-th input data $x_i$ with its corresponding label $y_i$, where $y_i \in [1 \ldots C]$ and $C$ is the class number. Define the j-th class anchor vector in softmax classifier $W_j$, where $j \in [1 \ldots C]$. The output feature of deep model is defined as $X_i$. 


Areas of different classes

Figure 2: Visualization of the learned features optimized by original Softmax vs. Virtual Softmax on MNIST dataset. We provide two cases of 2-D and 3-D visualization, which are shown in the left two columns and the right two columns, respectively. From the visualization, it can be observed that our Virtual Softmax possesses a stronger power of forcing compact and separable features.

In standard softmax, the optimization objective is to minimize the following cross-entropy loss:

\[ L_i = -\log \frac{e^{W_i^T X_i}}{\sum_{j=1}^{C} e^{W_j^T X_i}} \]  \hspace{1cm} (1)

here we omit the basis \( b \) and it does not affect the performance. Obviously, in original softmax optimization, it is to force \( W_i^T X_i > W_j^T X_i \) for all \( i \neq j \), in order to correctly classify the input \( x_i \).

Thus we can obtain the following property of softmax, which is the immediate incentive of our Virtual Softmax.

**Property 1.** As the number of class anchor vector increases, the constrained region for each class (e.g. the area of each class in Fig.4) becomes more and more compact, i.e. \( \theta \) will gradually decreases (e.g. in Fig.4).

**Proof.** For simplicity, here we consider the 2-D c-class case (n-D c-class case can be generalized as well but is more complicated). First, assume \( ||W_j|| = l, \forall j \) and they are evenly distributed in feature space, in another word, every two vectors have the same vectorial angle \( \Phi = \frac{2\pi}{c} \). As stated above, the softmax is to force \( W_i^T X_i > \max_{j \in c,j \neq y_i} (W_j^T X_i) \) in order to correctly classify \( x_i \), i.e. \( l \parallel X_i \parallel \cos \theta_{y_i} > \max_{j \in c,j \neq y_i} (l \parallel X_i \parallel \cos \theta_{j}) \Rightarrow \cos \theta_{y_i} > \max_{j \in c,j \neq y_i} (\cos \theta_{j}) \), where \( \theta_j \) denotes the angle between feature vector \( X_i \) and class vector \( W_j \). Hence, the decision boundary for class \( y_i \) is \( W_{y_i}^T X = \max_{j \in c,j \neq y_i} W_j^T X \), i.e. \( \cos \theta_{y_i} = \max_{j \in c,j \neq y_i} (\cos \theta_{j}) \), we define \( \theta \) as the angle between \( W_{y_i} \) and the decision boundary of class \( y_i \), and the solution is \( \theta = \theta_{y_i} = \arg \max_{j \in c,j \neq y_i} (W_j^T X) = \frac{\Phi}{2} \) (see Fig.1 (a) for a toy example). Considering the distribution symmetry of \( W \), the angle range of the constrained region for class \( y_i \) is \( 2\theta = \frac{2\pi}{c} \), which is inversely proportional to \( c \). Thus, if the class number increases, the constrained region of each class will be more and more compact. Moreover, for the \( ||W_{y_i}|| \neq ||W_j|| \) and non-uniform distribution case, the analysis is a little more complicated. Because the length of \( W \) are different, the feasible angles of class \( y_i \) and class \( j \) are also different (see the decision boundary of the case before injecting new classes in Fig.3(a)). Normally, the larger \( ||W_{y_i}|| \) is, the larger the feasible region of the class \( y_i \) is (22). While, as illustrated in Fig.3(b), if the class number increases, i.e. more new classes are injected to the same space, the constrained region of each class is going to be compact as well. \( \square \)

4 Virtual Softmax

Naturally motivated by the property[1], a simple way to introduce a large margin between classes for achieving intra-class compactness and inter-class separability is to constrain the features with more and more additional classes. In another word, injecting additional classes is to space the original classes, resulting in a margin between these original classes. Concretely, for a given classification task (namely that the number of categories to be classified is fixed), the additionally injected classes...
(if inserted at the exact position) will introduce a new and more rigorous decision boundary for these original classes, compressing their intra-class distribution, and furthermore the originally overlapped decision boundaries of the adjacent categories will thus be forced to separate with each other, producing a margin between these decision boundaries and thus encouraging inter-class separability (see Fig. 1 for a toy example). Therefore, injecting extra classes for feature supervision is a bold yet reasonable idea to enhance the discriminative property of features, and the objective function can be formulated as:

$$L_i = - \log \frac{e^{W^T y_i x_i}}{\sum_{j=1}^{C+K} e^{W^T y_i x_i}} = - \log \frac{e^{\|W_{y_i}\| \|X_i\| \cos \theta_j}}{\sum_{j=1}^{C+K} e^{\|W_j\| \|X_i\| \cos \theta_j}}$$

(2)

where $K$ is the number of extra injected classes. From Eq. 2, it can be observed that it is more possible to acquire a compact region for class $y_i$ quantified by the angle $\theta_{y_i}$, since the angle $\theta_{y_i}$ is optimized above a much larger set (C+K classes).

Theoretically, the larger $K$ is, the better features are. Nevertheless, in practical cases where the available training samples are limited and the total number of classes is changeless, it is intractable to enlarge the angular margin with real and existed extra classes. Moreover, a most troublesome problem is that we cannot insert the extra classes exactly between the originally adjacent categories. To address the aforementioned issues, we introduce a single and dynamic negative class into original softmax. This negative class is constructed on the basis of current training instance $x_i$. Since there is no real training data belonging to this class and it is employed only as a negative category (i.e. this class is utilized only to assist the training of original $C$ classes and have no need to be treated as a positive class), we denote it as virtual negative class and have the following formulation of our proposed Virtual Softmax:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i = - \frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{W^T y_i x_i}}{\sum_{j=1}^{C+K} e^{W^T y_i x_i}}$$

(3)

where $W_{virt} = \frac{\|W_{y_i}\|}{\|X_i\|}$ and $N$ is the training batch size. In the above equation, we instead require $W^T y_i x_i \geq \max (W^T y_i x_i, W^T y_i x_i, W^T y_i x_i)$, it is a special case of Eq. 2 where $K = 1$. Replacing a large and fixed set of virtual classes (like $K$ classes in Eq. 2) with a single dynamic virtual class $W_{virt}$ incurs nearly zero extra computational cost and memory consumption compared to original softmax.

From Eq. 3 one can observe that this virtual class is tactfully inserted around the class $y_i$, and particularly it is inserted at the same position with $X_i$ as illustrated in Fig. 2(a). It well matches our motivation that insert negative class between the originally adjacent classes $W_{y_i}$ and $W_j$, and this negative class will never stop pushing $X_i$ towards $W_{y_i}$ until they overlap with each other due to the dynamic nature of $X_i$ during training procedure. Moreover, from another optimization perspective, in order to correctly classify $x_i$, Virtual Softmax forces $W^T y_i x_i$ to be the largest one among $(C+1)$ inner product values. Since $W^T y_i x_i = \|W_{y_i}\| \|X_i\|$, the only way to make $W^T y_i x_i \geq \max_{j \in C+1}(W^T y_i x_i) = W^T y_i x_i$, i.e. $\|W_{y_i}\| \|X_i\| \cos \theta_{y_i} \geq \|W_{y_i}\| \|X_i\|$, is to optimize $\theta_{y_i} = 0$, however, the original softmax is to optimize $\|W_{y_i}\| \cos \theta_{y_i} \geq \max_{j \in C}(\|W_j\| \cos \theta_j)$, i.e. $\theta_{y_i} \leq \min_{j \in C}(\arccos (\|\|W_j\| \cos \theta_j\|))$ (briefly, if $\|W_j\|$ have the same magnitude, the original softmax is to optimize $\theta_{y_i} \leq \min_{j \in C}(\theta_j)$). Obviously, the original softmax is to make $\theta_{y_i}$ to be smaller than a certain value, while our Virtual Softmax optimizes a much more rigorous objective, i.e. zero limit $\theta_{y_i}$, aiming to produce more compact and separable features. And based on this optimization goal, the new decision boundary of class $y_i$ is overlapping with the class anchor $W_{y_i}$, which is more strict than softmax.

**Optimization:** The Virtual Softmax can be optimized with standard SGD and BP. And in backward propagation, the computation of gradients are listed as follows:

$$\frac{\partial L_i}{\partial X_i} = \sum_{j=1}^{C} e^{W^T j x_i} W_j + e^{W^T virt x_i} W_{virt} - W_{y_i} \frac{\partial L_i}{\partial W_{y_i}} = \frac{e^{W^T y_i x_i} X_i}{\sum_{j=1}^{C} e^{W^T j x_i} + e^{W^T virt x_i}} - X_i$$

(4)

Assume that all the $K$ classes are injected at the exact position among original $C$ classes.
Virtual Softmax

\[ \frac{\partial L_i}{\partial W_j} = \sum_{j=1}^{C} e^{W_{ij}^T X_i} + e^{W_{virt}^T X_i}, \text{ where } j \neq y_i \]

5 Discussion

Except for the above analysis that Virtual Softmax optimizes a much more rigorous objective than the original Softmax for learning discriminative features, in this section, we will give some interpretations from several other perspectives, i.e. coupling decay\(^{5.1}\) and feature update\(^{5.2}\). And we also provide the visualization of the learned features for intuitive understanding\(^{5.3}\).

5.1 Interpretation from Coupling Decay

In this paragraph, we give a macroscopic analysis from coupling decay which can be regarded as a regularization strategy.

Observing Eq.3, \( L_i \) can be reformulated as: \( L_i = -W_{yi}^T X_i + \log \left( \sum_{j=1}^{C} e^{W_{ij}^T X_i} + e^{W_{virt}^T X_i} \right) \), then performing the first order Taylor Expansion for the second term, a term of \( \|W_{yi}\| \|X_i\| \) shows up. Therefore, minimizing Eq.3 is to minimize \( \|W_{yi}\| \|X_i\| \) to some extend, and it can be viewed as a coupling decay term, i.e. data-dependent weight decay and weight-dependent data decay. It regularizes the norm of both feature representation and the parameters in classifier layer such that improves the generalization ability of deep models by reducing over-fitting. Moreover, it is supported by some experimental results, e.g. the feature norm in Fig.2 is decreased by Virtual Softmax (e.g. from 100 to 50 in 2-D space), and the performance improvement over the original Softmax is increased when using a much wider network as in Sec.6.1 (e.g. in CIFAR100/100+, increasing the model width from \( t=4 \) to \( t=7 \), the performance improvement is rising, since increasing the dimensionality of parameters while keeping data set size constant calls for stronger regularization).

However, the reason of calling the above analysis a macroscopic one is that it coarsely throws away many other relative terms and only considers the single effect of \( \|W_{yi}\| \|X_i\| \), without taking into account the collaborative efforts of other terms. Thus there are some other phenomena that cannot be explained well, e.g. why the inter-class angular margin is increased as shown in Fig.2 and why the confusion matrix of Virtual Softmax in Fig.5 shows more compact intra-class distribution and separable inter-class distribution than the original Softmax. To this end, we provide another discussion below from a relatively microscopic view that how the individual data representations of Virtual Softmax and original Softmax are constrained and formed in the feature space.

5.2 Interpretation From Feature Update

Here, considering the collaborative effect of all the terms in Eq.3, we give another microscopic interpretation from the perspective of feature update, which is also a strong justification of our method. In part, it reveals the reason why the learned features by Virtual Softmax are much more compact and separable than original softmax.

To simplify our analysis, we only consider the update in a linear feature layer, namely that the input to this linear feature layer is fixed. As illustrated in Fig.4(b), the feature vector \( X \) is computed as \( X = w^T Z \), where \( Z \) is the input vector, \( w \) is the weight parameters in this linear layer. We denote the \( k \)-th element of \( X \) as \( X_k \), the connected weight vector (i.e. the \( k \)-th column of \( w \)) as \( w_k \). Thus, \( X_k = w_k^T Z \), after computing the partial derivative \( \frac{\partial L}{\partial w_k} \), the parameters are updated by SGD as

\[ W_j = \sum_{i=1}^{C} e^{W_{ij}^T X_i} + e^{W_{virt}^T X_i}, \text{ where } j \neq y_i \]

\[ L_i = -W_{yi}^T X_i + \log \left( \sum_{j=1}^{C} e^{W_{ij}^T X_i} + e^{W_{virt}^T X_i} \right) \]

\[ \frac{\partial L}{\partial w_k} = \sum_{i=1}^{C} e^{W_{ij}^T X_i} X_i - X_k \]

\[ \frac{\partial L}{\partial w_k} = \sum_{i=1}^{C} e^{W_{ij}^T X_i} X_i - X_k \]
Although it is based on an approximate analysis, to some extent, it can give us an heuristic inter-


collaborative effort of many terms, we could know the working principle of our Virtual Softmax

can consistently improve the performances in both 2-D (99.2% vs. 98.91%) and 3-D (99.38% vs. 99.13%) cases. Furthermore, we also visualize the learned features in k-D space (where k > 3) with the confusion matrix. The confusion matrix comparison between original softmax and our Virtual Softmax are shown in

Specifically, we compute the included angle cosine (i.e. $\cos \theta$) between any two feature

\[ w'_k = w_k - \alpha \frac{\partial L}{\partial w_k}, \] 
\[ X'_k = (w'_k)^{T}Z = (w_k - \alpha \frac{\partial L}{\partial w_k})^{T}Z = w_k^{T}Z - \alpha (\frac{\partial L}{\partial w_k})^{T}Z = X_k - \alpha (\frac{\partial X_k}{\partial w_k})^{T}Z \]
\[ = X_k - \alpha \frac{\partial L}{\partial X_k} \] 
thus from Eq. 6, it can be inferred that the holistic feature vector $X'$ can be obtained by $X' = X - \beta \frac{\partial L}{\partial X}$, where $\beta = \alpha ||Z||^2$, implying that updating weight parameters $w$ with SGD can implicitly lead to the update of the output feature in the similar way. Based on this observation, putting the partial derivatives $\frac{\partial L}{\partial X}$ of Softmax and Virtual Softmax into Eq. 6 respectively, we can obtain the following corresponding updated features for Softmax (Eq. 7) and Virtual Softmax (Eq. 8) respectively:
\[ X' = X + \beta(W_{yi} - \sum_{j=1}^{C} e^{W_{ji}^{T}X}) \] 
\[ X' = X + \beta(W_{yi} - \sum_{j=1}^{C} e^{W_{ji}^{T}X} W_{j} + e^{W_{virt}^{T}X} W_{virt}) \]

Since the above equations are complicated enough to discuss, here for simplicity, we will give an approximate analysis. Consider a well trained feature (i.e. $e^{W_{virt}^{T}X} \gg e^{W_{j}^{T}X}$ and $e^{W_{yi}^{T}X} \gg e^{W_{j}^{T}X}$, $j \neq yi$) and that the parameters in Softmax and Virtual Softmax are the same. Then, omitting the relatively smaller terms in denominators, i.e. $\sum_{j=1,j\neq yi}^{C} e^{W_{ji}^{T}X} W_{j}$, Eq. 7,8 can be separately approximated as:
\[ X' = X + \beta(W_{yi} - \frac{e^{W_{yi}^{T}X} W_{yi}}{\sum_{j=1}^{C} e^{W_{ji}^{T}X}}) \] 
\[ X' = X + \beta(W_{yi} - \frac{e^{W_{yi}^{T}X} W_{yi}}{\sum_{j=1}^{C} e^{W_{ji}^{T}X} + e^{W_{virt}^{T}X}}) \]

from the above Eq.9 and Eq.10 we consider the magnitude and direction of 'vector1', 'vector2' and 'vector3', and then the updated $X'$ by the corresponding softmax and Virtual Softmax can be easily illustrated as in Fig.4(c). It can be firstly observed that the norm of red arrow is smaller than that of blue arrow, showing a similar result with the analysis in Sec.5.1, i.e. Virtual Softmax will regularize the feature norm. Meanwhile, one can also observe another important phenomenon that the feature vector $X'$ optimized by Virtual Softmax has a much smaller angle $\theta$ to the class anchor $W_{yi}$ than original Softmax, well explaining the reason why the features learned by Virtual Softmax is much more compact and separable than original Softmax. In summary, by considering the collaborative effort of many terms, we could know the working principle of our Virtual Softmax better. The Virtual Softmax not only provides regularization but more importantly, intensifies the discrimination property within the learned features.

Although it is based on an approximate analysis, to some extent, it can give us an heuristic interpretation of Virtual Softmax why it is capable of encouraging the discrimination of features from a novel feature update perspective. And practically, without these assumptions the Virtual Softmax can indeed produce discriminative features, validated by the visualization in Sec. 5.3 and the experimental results in Sec. 6.

5.3 Visualization of Compactness and Separability

In order to highlight and emphasize that Virtual Softmax indeed encourages the discriminative feature learning, we provide a clear visualization of the learned features on MNIST dataset [17] in 2-D and 3-D space respectively, as shown in Fig.5. From it, one can observe that the learned features by our Virtual Softmax are much compact and well separated, with larger inter-class angular margin and tighter intra-class distribution than softmax, and Virtual Softmax can consistently improve the performances in both 2-D (99.2% vs. 98.91%) and 3-D (99.38% vs. 99.13%) cases. Furthermore, we also visualize the learned features in k-D space (where $k > 3$) with the confusion matrix. The confusion matrix comparison between original softmax and our Virtual Softmax are shown in Fig.5. Specifically, we compute the included angle cosine (i.e. $\cos \theta$) between any two feature
We evaluate our Virtual Softmax on classification tasks and on face verification tasks. For fair
comparison, we train with the same network for both Virtual Softmax and the baseline softmax.

<table>
<thead>
<tr>
<th>Layer</th>
<th>MNIST (for fig.2)</th>
<th>MNIST</th>
<th>SVHN</th>
<th>CIFAR10</th>
<th>CIFAR100/100+</th>
</tr>
</thead>
</table>

Table 1: Model architectures for different benchmarks. [3x3, 32] denotes 4 cascaded convolutional layers
with 32 × t filters of size 3x3. The toy models are of t = 1. Max-Pooling is with 3 × 3 kernel and stride of 2.

![Figure 5: Cosine confusion matrices on the test splits of CIFAR10/100/100+. On each dataset, both Softmax
and Virtual Softmax use the same CNN architecture.](image)

vectors which are extracted from the testing splits of CIFAR10/100/100+ datasets. From Fig.5, it can be observed that, over the three datasets, the intra-class similarities are enhanced and the inter-class similarities are reduced when training with our Virtual Softmax. Moreover, our Virtual Softmax improves nearly 1%, 2% and 1.5% classification accuracies over the original softmax on CIFAR10/100/100+, respectively. To summarize, all these aforementioned experiments demonstrate that our Virtual Softmax indeed serves as an efficient algorithm which has a much stronger power of enlarging inter-class angular margin and compressing intra-class distribution than softmax and thus can significantly improve the recognition performances.

5.4 Relationship to Other Methods

There are a few research works concentrating on refining the Softmax objective, e.g. L-Softmax, A-Softmax and Noisy Softmax. Research works require to manually select a tractable constraint, and need a careful and annealing-like training procedure which is under human control. However, our Virtual-Softmax can be easily trained end-to-end without or with little human interventions, since the margin constraint is introduced automatically by the virtual class. Moreover, our method differs from L-Softmax and A-softmax in a heuristic idea that training deep models with the additional virtual class not only the given ones, which is the first work to extend the original Softmax to employ virtual class for discriminative feature learning and may inspire other researchers. And it can also provide regularization for deep models implicitly. Method aims to improve the generalization ability of DCNN by injecting noise into Softmax, which heads from a different perspective. In summary, our Virtual-Softmax comes from a clear and unusual motivation that injects a dynamic virtual class for enhancing features, which is different from the listed other methods, even though all the methods intend to learn better features.

6 Experiments and Results

We evaluate our Virtual Softmax on classification tasks and on face verification tasks. For fair
comparison, we train with the same network for both Virtual Softmax and the baseline softmax.

Small-Set Object Classification: We follow to devise our CNN models. Denote t as the
widening factor, which is used to multiply the basic number of filters in one convolutional layer, then
our architecture configurations are listed in Table 1. For training, the initial learning rate is 0.1, and is
divided by 10 at (20k, 27k) and (12k, 18k) in CIFAR100 and the other datasets respectively, and the
corresponding total iterations are 30k and 20k.

Fine-grained Object Classification: We fine-tune some popular models pre-trained on
ImageNet, including GooglenetV1 and GooglenetV2, by replacing the last softmax
layer with our Virtual Softmax. The learning rates are fixed to 0.0001 and 0.001 for the pre-trained
layers and the randomly initialized layer respectively, and stop training at 30k iteration.

Large-Set Object Classification: Here, we use the network for CIFAR100 with t=7, the learning
rate starts from 0.1 and is divided by 10 at 20k, 40k, 60k iterations. The maximal iteration is 70k.

Face Verification: We employ the published Resnet model in to evaluate our Virtual Softmax.
Start with a learning rate of 0.1, divide it by 10 at (30k, 50k) and stop training at 70k.
Compared Methods: We set L-Softmax (LS) [22], Noisy-Softmax (NS) [3] and A-Softmax (AS) [21] as the compared methods and re-implement them with the same experimental configurations as us. All of our experiments are implemented by Caffe [14]. The models are trained on one Titan X and we fill it with different batch sizes for different networks. For data preprocessing, we follow NS [3]. For testing, we use original softmax to classify the testing data in classification tasks and cosine distance to evaluate the performances in face verification tasks.

6.1 Ablation Study on Network Width

As listed in Table [1] our toy network on each dataset is with the widening factor of $t = 1$. We expand the toy models by setting $t = 1, 2, 3, 4$ and $t = 2, 3, 4, 7$ on CIFAR10 and CIFAR100 respectively, and the experimental results are listed in Table [2]. From these results, for example on CIFAR100+, it can be observed that as the model width increasing (i.e. from $t = 2$ to $t = 7$) the recognition error rate of Virtual Softmax is diminishing, and our method achieves consistent performance gain over the original softmax when training with different networks, verifying the robustness of our method. Furthermore, when training with Virtual Softmax, the recognition error rates of wider models are consistently lower than that of thinner models across all these datasets, indicating that the Virtual Softmax does not easily suffer from over-fitting and supporting the analysis in Sec.5.1. And we plot the curves of testing Acc on CIFAR10/100/100+ as shown in Fig [6], one can observe that our Virtual Softmax can be easily optimized, with a similar convergence speed compared to the original softmax.

6.2 Evaluation on object datasets

MNIST [17], SVHN [23], CIFAR [16] are the popular used small-set object classification datasets, including different number of classes. From Table [3] it can be observed that on both MNIST and SVHN datasets the Virtual Softmax not only surpasses the original softmax using the same network (i.e. 0.28% vs. 0.35% on MNIST, 1.93% vs. 2.11% on SVHN) but also outperforms LS [22], NS [3] and AS [21], showing the effectiveness of our method. Moreover, we also report the experimental results on CIFAR dataset as in Table [4]. Specifically, one can observe that our Virtual Softmax drastically improves nearly 0.6%, 2%, 1.5% accuracies over the baseline softmax on CIFAR10/100/100+ respectively. Meanwhile, it outperforms all of the other methods on both CIFAR10/100 datasets, e.g. it surpasses both the RestNet-110[9] and Densenet-40[11] on CIFAR100+ which are much deeper and more complex than our architecture, and also surpasses the listed compared methods.

CUB200 [35] is the popular fine-grained object classification set. The comparison results between other state-of-the-art research works and our Virtual Softmax are shown in Table [5] V1 and V2 denote the corresponding GoogleNet models. One can observe that the Virtual Softmax outperforms the baseline softmax over all the two pre-trained models, and surpasses all the compared methods LS [22], NS [3] and AS [21]. Additionally, training with only the Virtual Softmax, our final result is comparable to other remarkable works which exploit many assistant attention and alignment models, showing the superiority of our method.

<table>
<thead>
<tr>
<th>CIFAR10</th>
<th>$t = 1$</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>9.84</td>
<td>8.2</td>
<td>7.76</td>
<td>7.15</td>
</tr>
<tr>
<td>Virtual Softmax</td>
<td>8.95</td>
<td>7.27</td>
<td>7.04</td>
<td>6.68</td>
</tr>
<tr>
<td>Improvement</td>
<td>0.77</td>
<td>0.93</td>
<td>0.72</td>
<td>0.47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CIFAR100</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>32.72</td>
<td>30.74</td>
<td>28.76</td>
</tr>
<tr>
<td>Virtual Softmax</td>
<td>29.84</td>
<td>28.4</td>
<td>27.81</td>
</tr>
<tr>
<td>Improvement</td>
<td>2.88</td>
<td>2.34</td>
<td>0.95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CIFAR100+</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>30.44</td>
<td>28.59</td>
<td>27.21</td>
</tr>
<tr>
<td>Virtual Softmax</td>
<td>28.62</td>
<td>26.81</td>
<td>26.17</td>
</tr>
<tr>
<td>Improvement</td>
<td>1.82</td>
<td>1.78</td>
<td>1.94</td>
</tr>
</tbody>
</table>

Table 2: Recognition error rates(%) on CIFAR10/100/100+ datasets. $t$ is the widening factor and + denotes data augmentation.

Table 7: Acc (%) on ImageNet32
Acknowledgments: This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 61573068 and 61871052, Beijing Nova Program under Grant No. Z161100004916088, and sponsored by DiDi GAIA Research Collaboration Initiative.
References


