Constrained Center Loss for Convolutional Neural Networks

Zhanglei Shi, Hao Wang, and Chi-Sing Leung, *Senior Member, IEEE*

***Abstract*— From the feature representation's point of view, the feature learning module of a convolutional neural network (CNN) is to transform an input pattern into a feature vector. This feature vector is then multiplied with a number of output weight vectors to produce softmax scores. The common training objective in CNNs is based on the softmax loss, which ignores the intra-class compactness. This brief proposes a constrained center loss (CCL)-based algorithm to extract robust features. The training objective of a CNN consists of two terms, softmax loss and CCL. The aim of the softmax loss is to push the feature vectors from different classes apart. Meanwhile, the CCL aims at clustering the feature vectors such that the feature vectors from the same classes are close together. Instead of using stochastic gradient descent (SGD) algorithms to learn all the connection weights and the cluster centers at the same time. Our CCL-based algorithm is based on the alternative learning strategy. We first fix the connection weights of the CNN and update the cluster centers based on an analytical formula, which can be implemented based on the minibatch concept. We then fix the cluster centers and update the connection weights for a number of SGD minibatch iterations. We also propose a simplified CCL (SCCL) algorithm. Experiments are performed on six commonly used benchmark datasets. The results demonstrate that the two proposed algorithms outperform several state-of-the-art approaches.**

***Index Terms*— Center loss (CL), constrained center loss (CCL), convolutional neural networks (CNNs), image classification.**

NOMENCLATURE

Symbol	Definition
$\boldsymbol{x}$	Input of a CNN.
$\boldsymbol{x}_i$	ith training sample.
$y_i \in \{1, \ldots, C\}$	Class label of the ith training sample.
Θ	Collection of weights of the feature learning module of a CNN.
$\boldsymbol{f} \in \mathbb{R}^d$	Feature vector. This is the outputs of the feature learning module of a CNN. Note that $\boldsymbol{f}$ depends on Θ and $\boldsymbol{x}$.
d	Number of features (size of feature vector).
$\boldsymbol{f}_i$	Feature vector for the ith training sample. Note that $\boldsymbol{f}_i$ depends on Θ and $\boldsymbol{x}_i$.
$\boldsymbol{w}_k$	Output weight for the kth class.
$\boldsymbol{c}_k$	Feature center for the kth class.

I. INTRODUCTION

In pattern recognition, learning compact and separable features are crucial to improve the classification rate [1]–[6]. Over the years, many schemes were proposed for feature learning, such as linear discriminant analysis (LDA) [3], [4] and neural network (NN) [5], [6]. Those algorithms map the high-dimension input data on a low-dimensional space and thereby improve the computation efficiency. In addition, the intrinsic structure of the data is revealed. Hence, the resultant abstract features boost the classification accuracy. The classical LDA approach is to maximize the ratio of the between-class scatter and the within-class scatter. In the classical LDA for image data, images must be converted to vector form. Hence, some image spatial information may be lost and the computational complexity is high [4]. Therefore, some LDA works were proposed by directly handling images in matrix form, including the compound rank-k projection method [4] and convolutional 2-D nonlinear discriminant analysis method [7].

The convolutional neural network (CNN) model is a representative method for multiple levels of representations [8]–[13]. Unlike LDA, the success of the CNN model is beneficial from the utilization of spatial relationship of images. In CNN, the feature learning module is to transform an input pattern into a feature vector (deep features). This feature vector is then multiplied with a number of output weight vectors, each of which corresponds to a class. The classification process is then based on the softmax decision. By stacking more layers in a CNN, better robust features can be extracted. Representative CNN methods are stacked autoencoders [14], [15] and ResNet [16].

The softmax loss concept bundled with deep structures is popular in supervised learning [16]–[19]. It has two roles. First, it acts as classifier [20]. Second, it enlarges magnitudes of the features to boost its performance. This is because softmax can be considered as a softened max operator [21]. Hence, the deep features are pulled to fill the whole feature space [22]. However, the feature magnitudes of all classes are not magnified equally and the feature vectors from the same class would have imbalanced magnitudes [20], [23].

The extracted deep features of a trained CNN are expected to have high intra-similarity and low inter-similarity [20], [22], [24]–[27]. Unfortunately, raw data often suffer from low intra-class and relatively high inter-class similarity [10]–[13], [28]. To learn discriminative features, an alternative is metric learning [29]–[33]. The center loss (CL) algorithm and its variants [20], [23], [24], [34] aim at pushing the feature vectors from the same class together around a learnable class center. From prototype learning [35]–[37], the centers can be viewed as prototypes. Nevertheless, the update rules for centers in CL [20] prevent the learning of prototypes directly from data [35]. Besides, CL may lead to training instability [24]. To be more general than CL, the convolutional prototype learning (CPL) utilizes the distance-based cross-entropy loss (DCE) to learn the prototypes automatically from data [35]. To avoid overfitting, the DCE is equipped with the prototype loss (PL), leading to generalized convolutional prototype learning (GCPL) [35]. The combination of CNN and GCPL is also a case of convolutional prototype network (CPN) [36]. The linear combination of the classification loss (softmax/DCE) and distance-based penalization (CL/PL) gives a new perspective to learn robust features. The purpose of classification loss is to separate the features across classes, while the distance-based penalization aims at reducing the intra-class variance. In this way, both the intra-class compactness and inter-class separability can be improved. Compared to CL, GCPL completely relies on distance-based representation learning. It is noteworthy that GCPL becomes a linear classifier when the features and prototypes are normalized [37]. In this situation,

Manuscript received 19 October 2020; revised 16 May 2021; accepted 24 July 2021. Date of publication 24 August 2021; date of current version 6 February 2023. This work was supported in part by a Research Grant from City University of Hong Kong under Grant 7005223. *(Corresponding author: Chi-Sing Leung.)*

Zhanglei Shi and Chi-Sing Leung are with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong, SAR, China (e-mail: eeleungc@cityu.edu.hk).

Hao Wang is with the Guangdong Multimedia Information Service Engineering Technology Research Center, Shenzhen University, Shenzhen 518060, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3104392.

Digital Object Identifier 10.1109/TNNLS.2021.3104392

GCPL focuses on maximizing the cosine similarity between feature vectors and their corresponding prototypes.

Another pipeline is to enhance the cosine similarity between the feature vectors and the class weight vectors [21], [22], [26], [38], [39]. One approach is to perform the ℓ_2-normalization on feature vectors. It aims at eliminating the influence of imbalanced magnitudes. By adopting the ℓ_2-normalization, these algorithms integrate the ability to learn compact representations into softmax loss. Hence, one can explicitly reduce correlations among softmax weights and thereby increases the separation. Many frameworks, such as orthogonality constraint loss (OCL) [24], uniform loss [40] and others [41], are of this type. The OCL aims at orthogonalizing the softmax weights, whereas the uniform loss aims at pushing the weights to be eventually distributed. However, the ℓ_2-normalization in [22] and [26] only utilizes the unit direction of feature vectors. It does not normalize the feature vectors. The feature magnitudes still have large variation [22], [38], [39], [42]. This indicates that intra-class compactness still needs to be enhanced. The ℓ_2-normalization algorithms may introduce some normalization factors in the objective function and the computation of the gradient vector becomes complicated.

This brief proposes a constrained center loss (CCL)-based algorithm to improve intra-class similarity. In our algorithm, the CNN is jointly supervised by the softmax loss and CCL. The softmax loss aims at pushing the feature vectors from different classes apart, while the CCL aims at clustering the feature vectors from the same class together. Our algorithm uses the alternative training strategy that contains two steps. In the first step, we fix all the connection weights of the CNN and compute the class centers using all the samples. It is noteworthy that the center estimation step can be performed based on the minibatch concept. Afterward, we fix the class centers and then optimize other parameters based on the SGD concept for a number of minibatch iterations. The alternative procedure is repeatedly applied until all the estimation parameters converge. In addition, a simplified algorithm, namely simplified CCL (SCCL), is presented. Unlike some existing algorithms which involve normalization factors in the objective function, our algorithms remove some normalization factors. Compared to some state-of-the-art approaches, the proposed algorithms provide better performance on several benchmark datasets.

The rest of this brief is arranged as follows. Section II presents the backgrounds on feature representation, softmax loss, and CL. The proposed CLL and the SCCL algorithms are given in Section III. Section IV includes the experimental results. Finally, conclusions are presented in Section V.

II. Background

A. Feature Representation of CNN

Nomenclature summarizes the key notations in this brief. Other mathematical symbols are defined in their first appearances. Fig. 1 shows the feature representation concept. The feature learning module consists of a number of convolutional layers, a number of pooling layers, and a few fully connected layers. Let Θ be the collection of all weights in the feature learning module. The output layer is a fully connected layer. It consists of a number of output weight vectors, $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C\}$, where C is the number of classes, $\boldsymbol{w}_k \in \mathbb{R}^d$, and $k = 1, \ldots, C$. The feature learning module transforms an input pattern $\boldsymbol{x} \in \mathbb{R}^D$ (it may be a color image), to a feature vector $\boldsymbol{f} \in \mathbb{R}^d$. The feature vector is multiplied with the output weight vectors. The classification is then based on the softmax decision.

B. Softmax Loss

This brief considers that the training set contains N labeled pairs, given by $\mathcal{D} = \{\{\boldsymbol{x}_i, y_i\}_{i=1}^N\}$, where $\boldsymbol{x}_i$ is the ith training sample, its class label is denoted as $y_i \in \{1, \ldots, C\}$, and C is the number of classes. Given the ith sample $\boldsymbol{x}_i$, the output of the feature learning module is the corresponding feature vector, denoted by $\boldsymbol{f}_i$.

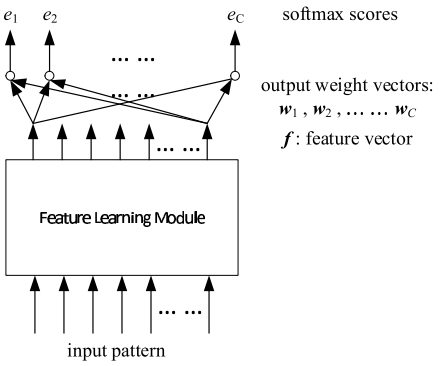

Fig. 1. Structure of a CNN.

Given an input $\boldsymbol{x}$, the CNN gives out the output softmax scores

$$p_{k'} = \frac{e_{k'}}{\sum_{k=1}^{C} e_k} = \frac{\exp\left(\boldsymbol{w}_{k'}^{\mathrm{T}} \boldsymbol{f} + b_{k'}\right)}{\sum_{k=1}^{C} \exp\left(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{f} + b_k\right)} \tag{1}$$

for $k' = 1, \ldots, C$, where $\boldsymbol{w}_k \in \mathbb{R}^d$ is the output weight vector for the kth class and b_k is the bias for the kth class.

For simplicity, we assume that all b_k's equal 0. Given a training input $\boldsymbol{x}_i$, its class label is denoted as y_i and the corresponding output weight vector for class y_i is denoted as $\boldsymbol{w}_{y_i}$. The softmax loss is written as

$$\mathcal{L}_{\text{softmax}} = -\sum_{i=1}^{N} \log \frac{\exp\left(\boldsymbol{w}_{y_i}^{\mathrm{T}} \boldsymbol{f}_i\right)}{\sum_{k=1}^{C} \exp\left(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{f}_i\right)}. \tag{2}$$

Also, we consider that the SGD with the minibatch concept is used to estimate all the weights of the CNN, including the weights of the feature learning module and the output weight vectors.

C. Softmax Loss Algorithms and Center Loss

The ℓ_2-normalization aims at reducing the variation in feature magnitudes. In [38], the optimization problem is given by

$$\min_{\boldsymbol{w}_k, \Theta} \mathcal{L}_{\text{softmax}}, \ \text{s.t.} \ \|\boldsymbol{f}_i\| = \alpha \quad \forall i = 1, \ldots, N \tag{3}$$

where α is a positive constant. The constrained optimization stated in (3) is difficult to solve because the feature vectors are functions of the connection weights of the feature learning module.

To avoid solving the highly nonlinear constrained optimization problem, stated in (3), the following normalization process is used. We feed the normalized feature vectors $\boldsymbol{f}_i/\|\boldsymbol{f}_i\|$'s to the output layer and scale them by α. The whole normalization process gives out $(\alpha \boldsymbol{f}_i/\|\boldsymbol{f}_i\|)$. It should be noted that the above normalization process does not really normalize the feature vectors [22], [38], [39]. With the feature normalization process, the problem [38] becomes an unconstrained optimization problem

$$\min_{\boldsymbol{w}_k, \Theta} -\sum_{i=1}^{N} \log \frac{\exp\left(\alpha \boldsymbol{w}_{y_i}^{\mathrm{T}} \dfrac{\boldsymbol{f}_i}{\|\boldsymbol{f}_i\|}\right)}{\sum_{k=1}^{C} \exp\left(\alpha \boldsymbol{w}_k^{\mathrm{T}} \dfrac{\boldsymbol{f}_i}{\|\boldsymbol{f}_i\|}\right)}. \tag{4}$$

The congenerous cosine loss (COCO) algorithm [22] and the von Mises–Fisher loss (vMF-A) algorithm [26] consider the normalization process on both the feature vectors and output weight vectors. Both the COCO and vMF-A algorithms aim at solving the following

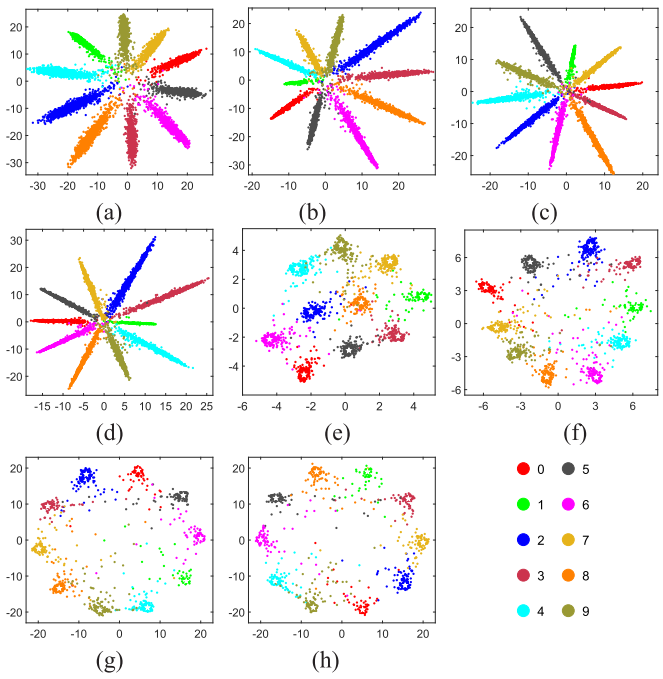

Fig. 2. Visualization of feature distribution and compactness learned by various algorithms on test set. The classes are distinguished by different colors. (a) Software. (b) COCO [22]. (c) OCL [24]. (d) vMF-A [26]. (e) GCPL [35]. (f) CL [20]. (g) CCL. (h) SCCL.

unconstrained optimization problem:

$$\min_{\boldsymbol{w}_k,\Theta} -\sum_{i=1}^{N} \log \frac{\exp\left(\alpha \dfrac{\boldsymbol{w}_{y_i}^{\mathrm{T}} \boldsymbol{f}_i}{\|\boldsymbol{w}_{y_i}\| \|\boldsymbol{f}_i\|}\right)}{\sum_{k=1}^{C} \exp\left(\alpha \dfrac{\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{f}_i}{\|\boldsymbol{w}_k\| \|\boldsymbol{f}_i\|}\right)}. \tag{5}$$

Again, as shown in Fig. 2, both the COCO and vMF-A algorithms do not really normalize the feature vectors. In the COCO and vMF-A algorithms, α is set to a constant, and the minibatch SGD learning is utilized to optimize the connection weights Θ of the feature learning module. The main difference between COCO and vMF-A is that COCO optimizes the output weight vectors $\boldsymbol{w}_k$'s by the minibatch concept, while vMF-A uses the batch model concept to update $\boldsymbol{w}_k$'s.

Since we have some normalization terms in (5), the gradient vector of the objective function has some terms with the factors $(1/\|\boldsymbol{w}_k\|)$'s and $(1/\|\boldsymbol{f}_i\|)$'s. This issue may lead to some computational problems when $\|\boldsymbol{w}_k\|$'s and $\|\boldsymbol{f}_i\|$'s are close to zero. Also, since $\boldsymbol{f}_i$'s are functions of the connection weights Θ of the feature learning module, the computation of the gradient vectors becomes complicated.

The CL approach [20] enhances the compactness of the feature vectors from the same class by penalizing the distances between the feature vectors and their centers. The CL is defined as

$$\mathcal{L}_{\text{intra}} = \frac{1}{2N} \sum_{i=1}^{N} \left\| \boldsymbol{f}_i - \boldsymbol{c}_{y_i} \right\|^2 \tag{6}$$

where $\boldsymbol{f}_i$ is the feature vector of the ith sample, y_i is the class label of the ith sample, and $\boldsymbol{c}_{y_i} \in \mathbb{R}^d$ is its class center.

The CL approach considers the following objective function:

$$\mathcal{L} = \mathcal{L}_{\text{softmax}} + \lambda \mathcal{L}_{\text{intra}} \tag{7}$$

where $\lambda > 0$ is the tradeoff parameter. In the CL approach, all the parameters are optimized by the SGD approach with the minibatch concept [20], [43].

III. Constrained Center Loss and Joint Supervision

Our models aim at learning a mapping that projects the inputs onto a deep feature space. In the feature space, the feature vectors should have high intra-class compactness and inter-class separability. In the following, we present our proposed algorithms.

A. Constrained Center Loss

Intra-class compactness [20] and feature normalization [22] can improve classification rate. In our formulation, we utilize both of them. To enhance the intra-class similarity, we consider the following optimization problem:

$$\min_{\Theta} \mathcal{L}_{\text{intra}}, \ \text{s.t.} \ \left\| \boldsymbol{f}_i \right\| = \alpha, \quad i = 1, \ldots, N \tag{8}$$

where

$$\mathcal{L}_{\text{intra}} = \frac{1}{2N} \sum_{i=1}^{N} \sum_{i' \neq i}^{N} \delta(y_i, y_{i'}) \left\| \boldsymbol{f}_i - \boldsymbol{f}_{i'} \right\|^2. \tag{9}$$

α is a positive constant and $\delta(y_i, y_{i'})$ is an indicator function. If $y_i = y_{i'}$, then $\delta(y_i, y_{i'}) = 1$. Otherwise, $\delta(y_i, y_{i'}) = 0$.

Even we use the minibatch concept with m samples in a minibatch, the complexity of computing the loss in (8) grows quadratically as the batch size m increases [22]. For example, we assume that each class consists of m/C samples in a minibatch. Hence, the total number of possible pairs is $(m^2 - mC/2C)$. Also, such loss stated in (8) results in slow convergence and training instability [19], [22], [39].

Inspired by CL [20], we introduce a class center for each class. Here, a class center can be viewed as the representative feature vector for a class. By using the class center concept, our formulation circumvents the difficulty of preparing the data pair. Instead of using hard constraints on the magnitudes of feature vectors, we consider the following constrained optimization task:

$$\min_{\boldsymbol{c}_k,\Theta} \frac{1}{2N} \sum_{i=1}^{N} \left\| \boldsymbol{f}_i - \boldsymbol{c}_{y_i} \right\|^2 \tag{10a}$$

$$\text{s.t.} \ \|\boldsymbol{c}_k\| = \alpha, \quad k = 1, \ldots, C. \tag{10b}$$

We call the loss in (10) as CCL. The term $\|\boldsymbol{f}_i - \boldsymbol{c}_{y_i}\|^2$ is distance between the feature vector $\boldsymbol{f}_i$ of the input pattern $\boldsymbol{x}_i$ and the center vector $\boldsymbol{c}_{y_i}$. By penalizing the distances $(1/2N) \sum_{i=1}^{N} \|\boldsymbol{f}_i - \boldsymbol{c}_{y_i}\|^2$, the feature vectors are pulled close to their corresponding centers [20], [35], [36]. In addition, we introduce constraints on the magnitudes of the center vectors. Hence, the feature vectors are softly normalized. When the training is finished, the magnitudes of feature vectors are roughly equal to α. In this way, we do not need to normalize the feature vectors by the ℓ_2-normalization process.

When $\boldsymbol{f}_i$'s and Θ are fixed, based on the Lagrange multiplier method, the class center for the kth class can be updated from the following equation:

$$\boldsymbol{c}_k = \alpha \frac{\sum_{i=1}^{N} \delta(k, y_i) \boldsymbol{f}_i}{\left\| \sum_{i=1}^{N} \delta(k, y_i) \boldsymbol{f}_i \right\|} \tag{11}$$

where $k = 1, \ldots, C$ and $\delta(k, y_i)$ is an indicator function. If $y_i = k$, then $\delta(k, y_i) = 1$. Otherwise, $\delta(k, y_i) = 0$. The derivation details of (11) can be found in Appendix I.

B. CCL Algorithm

We combine the CCL with the softmax loss $\mathcal{L}_{\text{softmax}}$ to formulate a joint supervision loss. Weight normalization in $\mathcal{L}_{\text{softmax}}$ is beneficial to maintain better angular classification margins among multiple classes. Hence, we assume that $\boldsymbol{w}_k$'s are normalized to obtain a better angular margin across classes. It should be noticed that the feature vectors do

Algorithm 1 CCL Approach

1: **repeat**
2: Fix all the connection weights of the CNN, and update centers $\{\boldsymbol{c}_k\}$ according to (11).
3: Fix centers, and train the CNN with SGD based on (13) for l minbatch presentations.
4: **until** convergence

not need to be normalized because the constrained loss given by (10) forces the magnitude of the feature vectors to be around α. Our joint supervision problem is given by

$$\min_{\boldsymbol{w}_k,\boldsymbol{c}_k,\Theta} -\sum_{i=1}^{N}\log\frac{\exp\left(\frac{\boldsymbol{w}_{y_i}^{\mathrm{T}}\boldsymbol{f}_i}{\|\boldsymbol{w}_{y_i}\|}\right)}{\sum_{k=1}^{C}\exp\left(\frac{\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{f}_i}{\|\boldsymbol{w}_k\|}\right)}+\frac{\lambda}{2N}\sum_{i=1}^{N}\|\boldsymbol{f}_i-\boldsymbol{c}_{y_i}\|^2 \quad (12a)$$

$$\text{s.t. } \|\boldsymbol{c}_k\|=\alpha,\quad k=1,\ldots,C \quad (12b)$$

where $\lambda>0$ is the tradeoff parameter that balances the importance of $\mathcal{L}_{\text{softmax}}$ and $\mathcal{L}_{\text{intra}}$.

In (12), there are three sets of parameters: the class centers $\boldsymbol{c}_k$'s, the connection weights Θ of the feature learning module, and the output weight vectors $\boldsymbol{w}_k$'s. We use the alternative strategy to optimize (12).

The alternative training process includes two steps and is repeated until the predefined termination condition reaches. Our approach is called the CCL algorithm. The first step updates the centers, whereas the second step updates the connection weights Θ of the feature learning module and the output weight vectors $\boldsymbol{w}_k$'s. We summarize the CCL approach in Algorithm 1.

1) Update the Centers: In the first step, we fix the connection weights Θ of the feature learning module and the output weight vectors $\boldsymbol{w}_k$'s. When all the connection weights of the CNN are fixed, a better idea is to seize the global distribution of the deep features by using the entire training samples. Hence, we can use (11) to update the centers. Note that in (11), the computation of the summation term can be implemented in a minibatch like manner. Specifically, we feed the minibatches to the network in a one-by-one manner and then compute the summation in the accumulated manner.

2) SGD Step: Update all the Connection Weights of CNN: In the second step, the centers are fixed and we update all the connection weights of the CNN based on the SGD concept. For the current minibatch, the loss is given by

$$\mathbb{L}=-\sum_{j=1}^{m}\log\frac{\exp\left(\frac{\boldsymbol{w}_{y_j}^{\mathrm{T}}\boldsymbol{f}_j}{\|\boldsymbol{w}_{y_j}\|}\right)}{\sum_{k=1}^{C}\exp\left(\frac{\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{f}_j}{\|\boldsymbol{w}_k\|}\right)}+\frac{\lambda}{2m}\sum_{j=1}^{m}\|\boldsymbol{f}_j-\boldsymbol{c}_{y_j}\|^2. \quad (13)$$

In this SGD step, we use the minibatch concept to minimize $\mathbb{L}$ by updating the connection weights Θ and $\boldsymbol{w}_k$'s. The SGD algorithm is run for l training minibatch iterations. Since $\mathbb{L}$ is differentiable, Θ and $\boldsymbol{w}_k$'s can be optimized by the minibatch SGD approach efficiently.

In the CCL algorithm, there are no the normalization factors, $(1/\|\boldsymbol{f}_j\|)$'s, in the objective function. However, the loss still has the normalization terms $(1/\|\boldsymbol{w}_k\|)$'s but $\boldsymbol{w}_k$'s are not functions of the connection weights Θ of the feature learning module.

C. Simplification of CCL

In the CCL algorithm, although one group of normalization factors is removed, the objective function still has the normalization factors: $(1/\|\boldsymbol{w}_k\|)$'s. The CCL algorithm forces $\boldsymbol{f}_i$'s to be close to both $\boldsymbol{w}_{y_i}$'s and $\boldsymbol{c}_{y_i}$'s in the angular distance manner. This indicates that $\boldsymbol{w}_{y_i}$ and $\boldsymbol{c}_{y_i}$ may have similar direction [21], [22]. Consequently, the coexistence of the classifiers and class centers in the CCL algorithm may result in some parameter redundancy.

To simplify the CCL algorithm, we propose the SCCL, in which the normalization factors, $(1/\|\boldsymbol{w}_k\|)$'s, are also removed. We use the directions of centers $\boldsymbol{c}_k$'s as the output weight vectors $\boldsymbol{w}_k$'s. The first step of the SCCL is the same as that of the CCL, that is, we update $\boldsymbol{c}_k$'s according to (11).

Afterward, the output weight vectors are obtained from $\boldsymbol{c}_k$'s by scaling a factor of $(1/\alpha)$. Therefore, the output weight vector $\boldsymbol{w}_k$ of the kth class is reduced to an unit direction, written as

$$\boldsymbol{w}_k=\frac{\boldsymbol{c}_k}{\alpha}=\frac{\sum_{i=1}^{N}\delta(k,y_i)\boldsymbol{f}_i}{\left\|\sum_{i=1}^{N}\delta(k,y_i)\boldsymbol{f}_i\right\|},\quad k=1,\ldots,C. \quad (14)$$

With this simplification, $C\times d$ parameters, i.e., the $\{\boldsymbol{w}_k\}_{k=1}^{C}\in\mathbb{R}^d$, are saved. Moreover, there is no normalization factor in the objective function of the SGD step.

Finally, the objective for the SGD step is

$$-\sum_{j=1}^{m}\log\frac{\exp\left(\frac{1}{\alpha}\boldsymbol{c}_{y_j}^{\mathrm{T}}\boldsymbol{f}_j\right)}{\sum_{k=1}^{C}\exp\left(\frac{1}{\alpha}\boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{f}_j\right)}+\frac{\lambda}{2m}\sum_{j=1}^{m}\|\boldsymbol{f}_j-\boldsymbol{c}_{y_j}\|^2. \quad (15)$$

In the SGD step, we only optimize the connection weights Θ of the feature learning module, and $\boldsymbol{c}_k$'s ($\boldsymbol{w}_k$'s) are not updated. The SGD algorithm is run for l training minibatch iterations. We summarize the SCCL algorithm in Algorithm 2.

Algorithm 2 Simplified CCL Approach

1: **repeat**
2: Fix the connection weights of the CNN, and update centers $\{\boldsymbol{c}_k\}$ according to (11).
3: Scale $\{\boldsymbol{c}_k\}$ with $\frac{1}{\alpha}$ to obtain the output weight vectors according to (14).
4: Fix the centers and output weight vectors, train CNN via the joint supervision (15) for l minbatch presentations.
5: **until** convergence

IV. Experiments

In this section, we first study the properties of our algorithms in Sections IV-B–IV-D. Afterward, we compare our algorithms with other comparison algorithms in Sections IV-E–IV-G.

We compare our proposed methods with the original softmax, vMF-A [26], COCO [22], CL [20], GCPL [35], and OCL [24]. Among them, the vMF-A and COCO are softmax loss such as algorithms. They normalize both $\boldsymbol{w}_k$ and $\boldsymbol{f}_i$ by the ℓ_2-normalization. Their objective functions are the same, as shown in (5). The network is trained to maximize the cosine similarity $(\boldsymbol{w}_{y_i}^{\mathrm{T}}\boldsymbol{f}_i/\|\boldsymbol{w}_{y_i}\|\|\boldsymbol{f}_i\|)$. The main difference between them is that vMF-A utilizes all the training samples to update classifiers $\boldsymbol{w}_k$'s. We integrate the OCL with COCO together and still denote it as OCL in the experiments.

A. Datasets and Settings

First, we test our methods on five visual classification datasets: CIFAR-10 [44], CIFAR-100 [44], Tiny ImageNet [45], Oxford-IIIT Pet [10], and Flower-102 [11]. The settings of training set and test set follow the common practice in these datasets. The CIFAR-10 dataset consists of 32×32 color images in ten classes, with 6000 images per class. There are 50 000 training images and 10 000 test images. The CIFAR-100 dataset includes 100 classes and has the same image resolution as in CIFAR-10. It contains 600 images per class,

of which 500 images are used for training and 100 images for testing. Tiny ImageNet dataset is a subset of ImageNet, including more than 100 000 images from 200 classes. For each class, there are 500 training images and 50 validation images. For the Flower-102 dataset and the Oxford-IIIT Pet dataset, we follow common practices [10], [11], [26].

Among those five datasets, the last two are fine-grained visual categorization (FGVC) datasets [10]–[13]. The FGVC case requires classifiers to distinguish the subcategory from the same supercategory. For instance, given a set of dog and cat images, conventional classification tasks aim at distinguishing the dogs (cats) from cats (dogs). In the FGVC case, the classifiers are required to classify the breeds of dogs or cats, such as Abyssinian (cat), Bengal (cat), Boxer (dog), and Chihuahua (dog). FGVC datasets are known that samples from the same subcategory share relatively low similarities, while samples of different subcategories often have high similarities. The correct recognition is determined by subtle and local features. Hence, the FGVC case is more challenging in comparison with general classification.

We choose ResNet [16] as the feature extractor for these five benchmark datasets. All models are implemented based on the Pytorch [46]. The momentum and weight decay for minibatch-based SGD are 0.9 and 0.0005, respectively. α is set to 40 for all experiments unless otherwise specified. In addition, no annotation information, such as bounding box or image segmentations, is used for FGVC datasets. In the comparison section, we run the experiments five times and report the results in "best accuracy, mean accuracy, and standard deviation" as in [16].

For CIFAR-10, CIFAR-100, and Tiny ImageNet, we train ResNet [16] from scratch with a batch size of 256. Considering the difficulty of the datasets, we use the ResNet34 [16] for Tiny ImageNet, and the ResNet18 is used for CIFAR-10 and CIFAR-100. For these three datasets, all algorithms are trained for 200 epochs. The learning rate starts with 0.1 and is divided by 10 every 50 epochs. We found that exploding gradients [47], [48] may happen in our approaches at the initial stage. The reason is that the pre-set $\alpha = 40$ may lead to large $\mathcal{L}_{\text{intra}}$ when the connection weights of the feature learning module of CNNs are randomly initialized. To avoid this problem, we adopt the warm-start technique [32]. Specifically, we use $\mathcal{L}_{\text{softmax}}$ to supervise the first several epochs.

For Oxford-IIIT Pet and Flower-102, we fine-tune ResNet34 [16] that is pretrained on ImageNet [45]. The batch size for these two datasets is 64. The learning rate begins with 0.001 for Oxford-IIIT Pet and 0.01 for Flower-102. It is divided by 10 after 80 and 120 epochs. The training is finished after 160 epochs. Here, the CCL approach and SCCL approach are directly applied to optimize the whole structure.

To further evaluate the proposed algorithms, we consider the face verification on labeled face in wild (LFW) dataset [9]. The LFW dataset contains 13 233 web-collected facial images from 5749 persons. The training set is CASIA-WebFace [49], including 494 414 images from 10 575 identities. For this task, we use a 20-layer CNN. The training lasts for 30 epochs with a batch size of 256. The learning rate starts from 0.1 and decreases by 0.1 after 16, 25 epochs.

B. Visualization of Feature Space

Before presenting the experimental results, we use the MNIST dataset [18] to visualize the feature vectors. Following the common practice in visualizing the distribution of feature vectors [20], [39], we set the dimension of the deep feature space of a small CNN to 2, i.e., $d = 2$. The CNN architecture is the same as that in [39]. In the example, we fix α at 20. The learned feature spaces of various algorithms are shown in Fig. 2. The figure shows the scattering plots of the extracted feature vectors. The horizontal axis is the first component of the feature vector, whereas the vertical axis is the second component.

TABLE I

CLASSIFICATION ACCURACIES (%) FOR VARIOUS NUMBERS OF MINIBATCH PRESENTATIONS IN THE SGD STEP. THE DATASET IS FLOWER-102. THE NUMBER OF RUNS IS FIVE. THE NETWORK STRUCTURE IS RESNET34

l	CCL			SCCL		
	best	mean	std	best	mean	std
16	95.56	95.37	0.26	95.56	95.35	0.18
32	95.46	95.34	0.10	95.54	95.23	0.23
48	95.46	95.30	0.16	95.33	<u>95.20</u>	0.10
64	**95.79**	95.50	0.17	95.37	95.24	0.09
96	95.77	95.65	0.09	95.41	95.28	0.12
128	95.67	95.58	0.08	95.59	95.38	0.15

softmax		
best	mean	std
93.48	93.24	0.22

We can observe that, compared to softmax, the COCO, OCL, and vMF-A algorithms lead to the situation that the feature vectors from the same class are with a "thinner" distribution in the feature space. This is because these three algorithms focus on learning a better angular distribution. Hence, the feature vectors are forced to be close to their class weight vectors in the angular distance. However, similar to the softmax loss, these three algorithms ignore the intra-class compactness and the features are still pulled to fill in the space, that is, large intra-class variation still exists.

On the contrary, the GCPL, CL, CCL, and SCCL algorithms improve the intra-class similarity. Feature samples are distributed around their class centers as shown in Fig. 2(e)–(h). In addition, the feature samples of CCL and SCCL are distributed on the surface of a hypersphere, that is, CCL and SCCL can provide a normalization effect on feature magnitude.

Note that in Section IV-G, we use the CIFAR-10 to show the cosine similarity of arbitrary pairs of feature vectors in various algorithms.

C. Effect of the Number l of Minibatch Training in the SGD Step

In our two proposed algorithms, we use the alternative scheme to update the network parameters. For the first step that updates the centers, we have an analytical formula. For each SGD step, we need to train the remaining parameters for l minibatch presentations.

To study the influence of l, we set $\lambda = 0.1$ and consider the Flower-102 and Oxford-IIIT Pet datasets with various l values. The values of l for Flower-102 and Oxford-IIIT Pet are $l \in \{16, 32, 48, 64, 96, 128\}$ and $l \in \{29, 58, 70, 87, 100, 116\}$, respectively. The results for Flower-102 and Oxford-IIIT Pet are reported in Tables I and II, respectively. For easier comparison, we also list the results of the softmax in the tables.

We can observe that the proposed CCL and SCCL algorithms are superior to the softmax algorithm on both datasets. For the Flower-102 dataset, the classification performances of the two proposed algorithms are higher than 95.0%, with the smallest mean accuracy 95.20% given by the SCCL ($l = 48$). The average accuracy of softmax is 93.24% that is around 2% lower than the average accuracies of CCL and SCCL. For the best result, the CCL algorithm achieves an accuracy of 95.79% when $l = 64$. The result is 2.31% higher than the best result (93.48%) of softmax.

Similarly, for the Oxford-IIIT Pet dataset, the accuracies of the two proposed algorithms are higher than 92.0%. The smallest mean accuracy is the case that the SCCL with $l = 70$. The average accuracy of softmax is 90.55%, which is around 2% lower than the average accuracies of CCL and SCCL. For the best result, the CCL

TABLE II

CLASSIFICATION ACCURACIES (%) FOR VARIOUS NUMBERS OF MINIBATCH PRESENTATIONS IN THE SGD STEP. THE DATASET IS OXFORD-IIIT PET. THE NUMBER OF RUNS IS FIVE. THE NETWORK STRUCTURE IS RESNET34

l	CCL			SCCL		
	best	mean	std	best	mean	std
29	92.83	92.68	0.13	92.48	92.36	0.09
58	92.70	92.67	0.04	92.50	92.37	0.16
70	**92.89**	92.66	0.20	92.29	<u>92.19</u>	0.06
87	92.72	92.47	0.20	92.59	92.30	0.17
100	92.80	92.61	0.21	92.48	92.31	0.12
116	92.75	92.67	0.05	92.40	92.31	0.10

softmax		
best	mean	std
91.03	90.55	0.37

TABLE III

CLASSIFICATION ACCURACIES (%) WITH VARIOUS LOSS WEIGHT λ'S ON THE CIFAR-10 DATASET OVER FIVE RUNS. THE NETWORK STRUCTURE IS RESNET18

λ	CCL			SCCL		
	best	mean	std	best	mean	std
0.01	94.61	94.50	0.10	**94.83**	94.59	0.18
0.02	94.60	94.52	0.06	94.75	94.51	0.14
0.05	94.56	94.45	0.14	94.45	94.35	0.10
0.1	94.55	94.38	0.15	94.67	94.38	0.29
0.2	94.38	94.22	0.11	94.44	94.36	0.09
0.5	94.26	<u>94.10</u>	0.12	94.64	94.47	0.13

softmax		
best	mean	std
93.81	93.57	0.17

TABLE IV

CLASSIFICATION ACCURACIES (%) WITH VARIOUS LOSS WEIGHT λ'S ON THE CIFAR-100 DATASET OVER FIVE RUNS. THE NETWORK STRUCTURE IS RESNET18

λ	CCL			SCCL		
	best	mean	std	best	mean	std
0.01	76.64	76.23	0.31	**77.28**	76.87	0.29
0.02	76.50	76.39	0.12	76.99	76.58	0.24
0.05	76.70	76.49	0.16	77.27	76.83	0.27
0.1	77.01	76.68	0.22	77.23	76.85	0.33
0.2	76.79	76.39	0.24	76.76	76.53	0.23
0.5	76.49	75.98	0.41	75.45	<u>75.07</u>	0.28

softmax		
best	mean	std
75.12	74.87	0.21

algorithm achieves 1.86% higher performance than 91.03% of softmax.

Overall, the proposed algorithms work well with a large range of l, though the centers are not updated in each training iteration of SGD. In addition, small standard deviations of the proposed algorithms indicate that they can provide stable classification results with various l's.

D. Classification Performance With Various Loss Weight λ

The weighting parameter λ can balance $\mathcal{L}_{\text{softmax}}$ and $\mathcal{L}_{\text{intra}}$. We use the CIFAR-10 and CIFAR-100 datasets to discuss the influence of λ. We try several $\lambda \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ for the two datasets. During training, the centers are updated every 196 minibatch presentations. The results for CIFAR-10 and CIFAR-100 are listed in Tables III and IV, respectively.

It is observed that our proposed structures consistently improve the performance of the original softmax. For CIFAR-10, CCL and SCCL can improve the accuracy from 93.57% to over 94.10% with 0.53% gain averagely. Improvements can also be achieved on CIFAR-100 from 74.87% to over 75.07% in average. The facts indicate that techniques related to intra-class compactness are needed to improve the feature discrimination. The weighting parameter can be selected over a wide range from 0.01 to 0.2. In this range, the performance variation is not very large. This means that good λ can be selected in an efficient way. However, λ cannot be set too large. For example, when $\lambda = 0.5$, the averaged results of both CCL and SCCL are lower than 76.0% on the CIFAR-100 dataset. Also, small standard deviation values of the proposed algorithms indicate that both CCL and SCCL work well and stable with various λ values.

E. Classification Performance Comparison

The comparison results are listed in Table V. For each dataset, the best result between CCL and SCCL is marked in bold, while the best result of their counterparts is marked in underline. From the table, our methods outperform CL on all datasets with large improvements. For CIFAR-10, CCL improves the performance of CL from 93.87% to 94.61%, which is 0.74% improvement. In addition, SCCL is 2.08% higher than CL for CIFAR-100 and 1.38% higher for Flower-102. CCL also achieves 1.81% and 1.58% improvements on these two datasets over CL. The accuracy of SCCL is 0.98% higher than 91.61% of CL on Oxford-IIIT Pet. On Tiny ImageNet, the SCCL and CCL improve the CL from 63.84% to 63.91% and 64.47%, respectively.

For the comparison among all methods, the accuracy of SCCL is 0.88% higher than that of COCO on CIFAR-10 and 2.08% higher than that of CL on CIFAR-100. For Flower-102, CCL improves the best result of vMF-A from 95.01% to 95.79%, with 0.78% gain. CCL achieves an accuracy of 92.89% on Oxford-IIIT Pet, which is 0.47% higher than the accuracy 92.42% of vMF-A. The CCL has 0.63% gain over CL on Tiny ImageNet from 63.84% to 64.47%.

From Table V, the proposed algorithms outperform their counterparts both in terms of best results and the averaged results on all five datasets.

F. Face Verification

Following the standard protocol of unrestricted with labeled outside data, we test 6000 pairs of face images. We extract the 512-D feature vector for each image and its horizontally flipped one and concatenate them as the final identity representation. We use the cosine similarity of two identity representations as the score for verification. The 6000 pairs are divided into ten folds. We compute the verification accuracies on them and report the average results in Table VI. Overall, CCL gives the highest verification accuracy of 99.00%, outperforming 98.72% of softmax and 98.77% of CL by 0.28% and 0.23%, respectively. In addition, CCL outperforms the COCO, vMF-A, GCPL, and OCL by 0.45%, 0.70%, 0.32%, and 0.55%, respectively.

G. Visualization of Intra-Class Similarity and Inter-Class Separability

In this section, we present the confusion matrices of various approaches [39]. The CIFAR-10 dataset is considered and the confusion matrices are shown in Fig. 3. In the figures, the lighter color means that the corresponding feature vectors are more similar (the cosine similarity is larger). On the contrary, darker color represents that the feature vectors have lower similarity. It can be observed that the matrices of our two algorithms are lighter on the main diagonal and darker in the other areas. Hence, the proposed approaches can significantly enhance the intra-class similarity.

To further investigate the intra-class similarity and inter-class separability, we record the histograms of the cosine similarities for

TABLE V

CLASSIFICATION ACCURACIES (%) OF VARIOUS ALGORITHMS ON FIVE BENCHMARK DATASETS OVER FIVE RUNS

Method	CIFAR-10			CIFAR-100			Flower-102			Oxford-IIIT Pet			Tiny ImageNet		
	best	mean	std	best	mean	std	best	mean	std	best	mean	std	best	mean	std
softmax	93.81	93.57	0.17	75.12	74.87	0.21	93.48	93.24	0.22	91.03	90.55	0.37	63.65	63.49	0.13
COCO [22]	93.95	93.86	0.12	74.60	74.20	0.30	94.63	94.28	0.34	92.34	92.12	0.19	63.09	62.84	0.16
vMF-A [26]	93.59	93.46	0.12	74.13	74.02	0.08	95.01	94.82	0.15	92.42	92.29	0.11	62.99	62.82	0.15
CL [20]	93.87	93.67	0.28	75.20	74.83	0.37	94.21	93.77	0.29	91.61	91.28	0.19	63.84	63.62	0.23
GCPL [35]	93.28	93.08	0.19	70.54	70.07	0.47	89.15	88.71	0.38	90.57	90.10	0.33	62.21	61.81	0.42
OCL [24]	93.92	93.72	0.19	74.66	74.38	0.27	94.94	94.84	0.16	92.23	92.03	0.20	63.41	62.97	0.36
CCL	94.61	94.50	0.10	77.01	76.68	0.22	**95.79**	95.50	0.17	**92.89**	92.66	0.20	**64.47**	64.28	0.15
SCCL	**94.83**	94.59	0.18	**77.28**	76.87	0.29	95.59	95.38	0.15	92.59	92.30	0.17	63.91	63.76	0.16

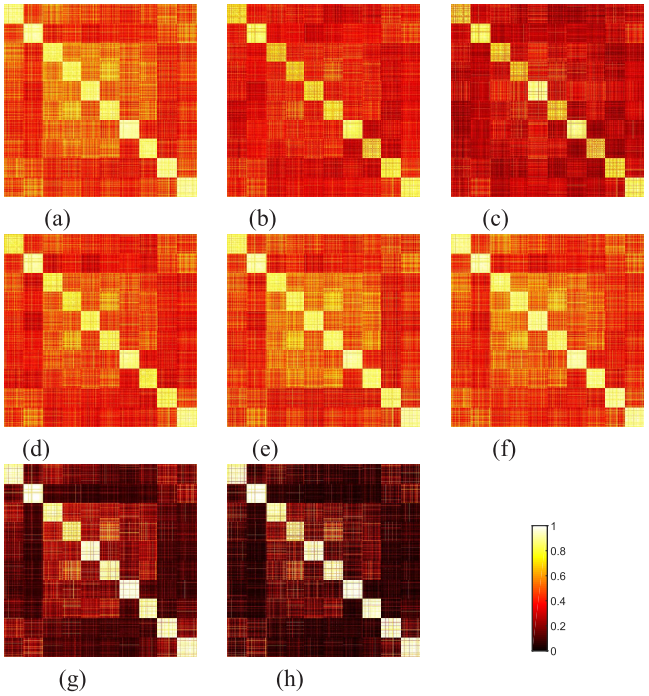

Fig. 3. Confusion matrices on the CIFAR-10 dataset. (a) Software. (b) COCO [22]. (c) OCL [24]. (d) vMF-A [26]. (e) GCPL [35]. (f) CL [20]. (g) CCL. (h) SCCL.

TABLE VI

FACE VERIFICATION PERFORMANCE (%) ON LFW

Method	Outside Data	best	mean	std
softmax	WebFace	98.72	98.65	0.07
COCO [22]	WebFace	98.55	98.36	0.16
vMF-A [26]	WebFace	98.30	98.18	0.07
CL [20]	WebFace	98.77	98.64	0.09
GCPL [35]	WebFace	98.68	98.58	0.08
OCL [24]	WebFace	98.45	98.28	0.11
CCL ($\alpha = 10$)	WebFace	**99.00**	98.87	0.10
SCCL ($\alpha = 10$)	WebFace	98.70	98.57	0.08

positive and negative pairs [22], [39]. A positive pair means that the two feature vectors are from the same class, whereas a negative pair means that the two feature vectors are from different classes.

When the feature vectors from different classes are well separated, the peak of the negative pair histogram should be near zero in terms of cosine similarity. On the other hand, when the feature vectors from the same class are well clustered, the peak of the positive pair histogram should be near one in terms of cosine similarity.

As shown in Fig. 4(g) and (h) compared to other algorithms, for our proposed algorithms, the peaks of the positive pair histograms

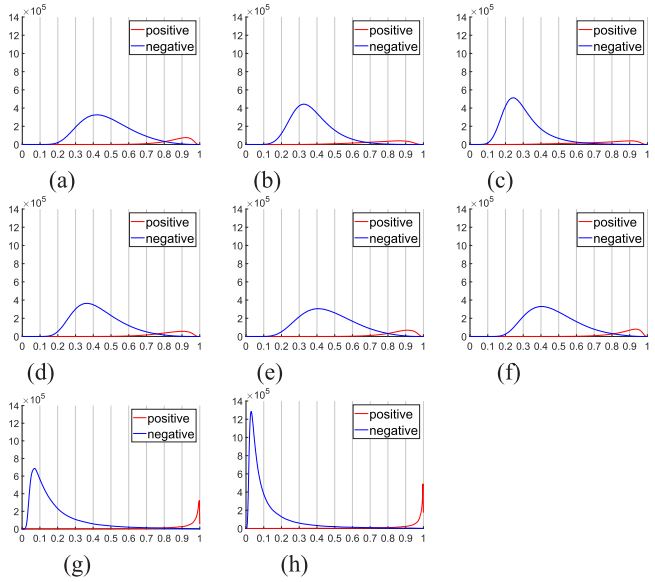

Fig. 4. Histogram of cosine similarity for positive and negative pairs on CIFAR-10 for different methods. (a) Software. (b) COCO [22]. (c) OCL [24]. (d) vMF-A [26]. (e) GCPL [35]. (f) CL [20]. (g) CCL. (h) SCCL.

TABLE VII

MEAN AND STANDARD DEVIATION OF COSINE SIMILARITY FOR POSITIVE AND NEGATIVE PAIRS ON THE CIFAR-10 DATASET

Method	Positive pairs		Negative pairs		Difference
	mean	std	mean	std	
softmax	0.8606	0.0850	0.4748	0.1382	0.3858
COCO [22]	0.7515	0.1387	0.3711	0.1131	0.3804
vMF-A [26]	0.8265	0.1023	0.4255	0.1311	0.4010
CL [20]	0.8692	0.0862	0.4643	0.1392	0.4049
GCPL [35]	0.8519	0.0896	0.4613	0.1454	0.3906
OCL [24]	0.7469	0.1635	0.2992	0.1136	0.4477
CCL	0.9023	0.1357	0.1848	0.1554	0.7175
SCCL	0.9006	0.1549	0.1252	0.1456	0.7754

are closer to one. This means that, with our algorithms, the feature vectors from the same class are well clustered.

At the same time, compared to other algorithms, for our proposed algorithms, the peaks of the negative pair histograms are closer to zero. This means that, with our algorithms, the feature vectors from different classes are well separated.

Table VII shows the cosine similarities for positive and negative pairs. In the table, the "Difference" is the difference between the mean similarity of positive pairs and the mean similarity of negative pairs. We can observe that, compared to other algorithms, our proposed algorithms have larger mean similarities for positive pairs. This indicates that our proposed algorithms can improve intra-class similarity. Meanwhile, our proposed algorithms have smaller mean similarities for negative pairs. In addition, our proposed algorithms

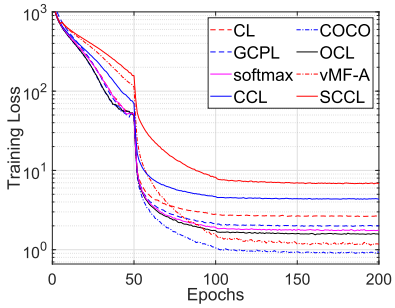

Fig. 5. Convergence on Tiny ImageNet.

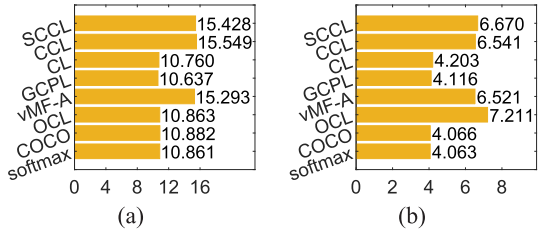

Fig. 6. Average training time (in hours) over five runs on (a) Tiny ImageNet and (b) CASIA-WebFace.

have larger differences between the mean similarities for negative pairs and the mean similarities for positive pairs. We can conclude that our proposed algorithms can enhance inter-class separability.

H. Convergence and Computational Complexity

First, we investigate the convergence speed of the different approaches. We show the training loss of all the algorithms on the Tiny ImageNet dataset in Fig. 5. Here, we update the centers at each epoch (not each minibatch iteration). It is observed that CCL and SCCL have a comparable convergence speed to other methods.

The computational complexity for backward propagation is about two times of forward propagation [50]. Let Γ be the complexity per image of forward propagation. Then, the total complexity of a training epoch is 3 $N\Gamma$, where N is the number of images in the training set.

In the CCL and SCCL algorithm, for every l minibatch presentations, we update the centers. When we update the centers, we need to perform a forward propagation. Hence, the complexity of the center update is $N\Gamma$. Let m be the size of a minibatch and τ be the number of center updates in an epoch. Clearly, we have $\tau = (N/ml)$. Hence, the total complexity of CCL and SCCL is $(3+\tau)N\Gamma$ for a training epoch. From our experience, τ can be chosen from 0.5 to 2. Note that $\tau = 0.5$ means that we update the center every two epochs. Here, we use the cases of $\tau \in \{1, 2\}$ to illustrate the complexities of CCL and SCCL.

For Tiny ImageNet, the complexity per image of a forward propagation [16] is 3.675×10^9 and we set $\tau = 1$. As there are 10^5 training images, the total complexity for an epoch without center update is $3 \times 3.675 \times 10^{14} = 1.103 \times 10^{15}$. For the proposed CCL and SCCL with $\tau = 1$, the complexity is then equal to 1.470×10^{15}. This means that the additional complexity is 3.675×10^{14}.

For CASIA-WebFace, the complexity per image of a forward propagation is 1.843×10^9 and we set $\tau = 2$. As there are around 0.5×10^6 training images, the total complexity for an epoch without center update is $1.5 \times 1.843 \times 10^{15} = 2.764 \times 10^{15}$. Thus, the complexity for CCL and SCCL is 4.608×10^{15}. This means that the additional complexity is 1.844×10^{15}.

To conclude, the computational complexity of the proposed algorithm and their counterparts remains in the same order.

Computational complexity of various methods can be verified by measuring the training time. We report the average training time of various algorithms over five runs in Fig. 6. It is observed that our algorithms and vMF-A take roughly 15.5 h on Tiny ImageNet and around 6.6 h on CASIA-WebFace. It is noteworthy that all algorithms have the similar inference time, and hence, we do not show the inference time. It is because the inference procedure of all algorithms is nearly the same. To conclude, the improved classification accuracy and efficient inference suggest that the proposed algorithms could be a candidate in real applications.

V. Conclusion

Intra-class compactness and inter-class separability of deep features are important to improve the CNN discrimination ability. This brief proposes the CCL and SCCL for enhancing the intra-class compactness. Combined with the softmax loss, the joint supervision scheme can maximize the compactness and separability simultaneously. Moreover, the two proposed algorithms learn class center analytically with utilizing the entire training set. In this way, the global information of the deep feature space is captured by our formulation. Based on the alternative strategy, we propose two learning algorithms to optimize the structure. The experiments are conducted on six commonly used benchmark datasets. It is demonstrated that compact and separate features can be extracted with the proposed CCL algorithm and SCCL algorithm. In addition, the proposed methods are better than several state-of-the-art approaches. Similar to CL and PL, the two proposed algorithms can reduce the intra-class variation and restrict the feature distribution to be around their class centers. Thus, it can preserve spaces for unknown classes. In the future, we will explore the class-incremental learning scenario, where the CNN needs to deal with unknown classes and maintain good performance for the known classes simultaneously.

Appendix I
Derivation of Center Update

Consider that the connection weights Θ of the feature learning module are fixed, and the centers in (10) can be solved as follows. The problem stated in (10) can be decomposed into C individually constrained optimization problems.

For simplicity, we focus on one specific Class y with center $\boldsymbol{c}_y$, where $y = 1, \ldots, C$. To maximize the similarities between the instances and their corresponding center, we have to minimize $\mathcal{L}_{\text{intra}_y}$ subject to $\boldsymbol{c}_y^{\mathrm{T}}\boldsymbol{c}_y = \alpha^2$.

Let N_y be the number of samples that belonged to Class y and let $\{\boldsymbol{f}_{i_y}\}$'s be the feature vectors of the training samples belonged to Class y. We introduce a Lagrange multiplier ζ. The Lagrangian of the objective function with respect to Class y is given by

$$\mathcal{L}(\boldsymbol{c}_y, \zeta) = \frac{1}{2N_y}\sum_{i_y=1}^{N_y}\left\|\boldsymbol{f}_{i_y} - \boldsymbol{c}_y\right\|^2 + \zeta\left(\alpha^2 - \boldsymbol{c}_y^{\mathrm{T}}\boldsymbol{c}_y\right). \tag{16}$$

Differentiating the Lagrangian function with respect to ζ and $\boldsymbol{c}_y$ and setting the derivatives to zero, we obtain

$$-\frac{1}{N_y}\sum_{i_y=1}^{N_y}\left(\boldsymbol{f}_{i_y} - \boldsymbol{c}_y\right) - 2\zeta\boldsymbol{c}_y = \boldsymbol{0} \text{ and } \boldsymbol{c}_y^{\mathrm{T}}\boldsymbol{c}_y = \alpha^2. \tag{17}$$

From (17), we get $\boldsymbol{c}_y = (1/(1-2\zeta)N_y)\sum_{i_y=1}^{N_y}\boldsymbol{f}_{i_y}$. Defining $\boldsymbol{r}_y = \sum_{i_y=1}^{N_y}\boldsymbol{f}_{i_y}$, we can obtain $(1/(N_y - 2N_y\zeta)^2)\boldsymbol{r}_y^{\mathrm{T}}\boldsymbol{r}_y = \alpha^2$. Hence, $\zeta = (1/2)(1 - (\|\boldsymbol{r}_y\|/\alpha N_y))$ and

$$\boldsymbol{c}_y = \frac{1}{(1-2\zeta)N_y}\boldsymbol{r}_y = \alpha\frac{\boldsymbol{r}_y}{\|\boldsymbol{r}_y\|} = \frac{\alpha\sum_{i_y=1}^{N_y}\boldsymbol{f}_{i_y}}{\left\|\sum_{i_y=1}^{N_y}\boldsymbol{f}_{i_y}\right\|}. \tag{18}$$

REFERENCES

[1] X. Chang, F. Nie, Y. Yang, C. Zhang, and H. Huang, "Convex sparse PCA for unsupervised feature learning," *ACM Trans. Knowl. Discovery Data*, vol. 11, no. 1, pp. 1–16, 2016.

[2] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A simple deep learning baseline for image classification?" *IEEE Trans. Image Process*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.

[3] M. Dorfer, R. Kelz, and G. Widmer, "Deep linear discriminant analysis," 2015, *arXiv:1511.04707*. [Online]. Available: http://arxiv.org/abs/1511.04707

[4] X. Chang, F. Nie, S. Wang, Y. Yang, X. Zhou, and C. Zhang, "Compound rank-k projections for bilinear analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 7, pp. 1502–1513, Jul. 2015.

[5] H. Li, Y. Liu, W. Ouyang, and X. Wang, "Zoom out-and-in network with map attention decision for region proposal and object detection," *Int. J. Comput. Vis.*, vol. 127, no. 3, pp. 225–238, 2019.

[6] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[7] Q. Wang, Z. Qin, F. Nie, and X. Li, "C2DNDA: A deep framework for nonlinear dimensionality reduction," *IEEE Trans. Ind. Electron.*, vol. 68, no. 2, pp. 1684–1694, Feb. 2020.

[8] Y. Sun, X. Wang, and X. Tang, "Deeply learned face representations are sparse, selective, and robust," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2892–2900.

[9] G. B. Huang and E. Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep. 4–003, 2014.

[10] O. M Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3498–3505.

[11] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2008, pp. 722–729.

[12] Y. Zhang, K. Jia, and Z. Wang, "Part-aware fine-grained object categorization using weakly supervised part detection network," *IEEE Trans. Multimedia*, vol. 22, no. 5, pp. 1345–1357, May 2020.

[13] Y. Cui, F. Zhou, Y. Lin, and S. Belongie, "Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1153–1162.

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[15] L. Meng, S. Ding, N. Zhang, and J. Zhang, "Research of stacked denoising sparse autoencoder," *Neural Comput. Appl.*, vol. 30, no. 7, pp. 2083–2100, Oct. 2018.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[17] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[19] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, and J. Zhu, "Rethinking softmax cross-entropy loss for adversarial robustness," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–19.

[20] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 499–515.

[21] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace: L_2 hypersphere embedding for face verification," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 1041–1049.

[22] Y. Liu, H. Li, and X. Wang, "Rethinking feature discrimination and polymerization for large-scale recognition," 2017, *arXiv:1710.00870*. [Online]. Available: http://arxiv.org/abs/1710.00870

[23] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A comprehensive study on center loss for deep face recognition," *Int. J. Comput. Vis.*, vol. 127, nos. 6–7, pp. 668–683, 2019.

[24] W. Wang, W. Pei, Q. Cao, S. Liu, G. Lu, and Y.-W. Tai, "Push for center learning via orthogonalization and subspace masking for person re-identification," *IEEE Trans. Image Process.*, vol. 30, pp. 907–920, 2021.

[25] Y. Luo, Y. Wong, M. Kankanhalli, and Q. Zhao, "$\mathcal{G}$-softmax: Improving intraclass compactness and interclass separability of features," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 685–699, May 2020.

[26] X. Zhe, S. Chen, and H. Yan, "Directional statistics-based deep metric learning for image classification and retrieval," *Pattern Recognit.*, vol. 93, pp. 113–123, Sep. 2019.

[27] L. Zhou, Z. Wang, Y. Luo, and Z. Xiong, "Separability and compactness network for image recognition and superresolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3275–3286, Nov. 2019.

[28] X. Li *et al.*, "Supervised latent Dirichlet allocation with a mixture of sparse softmax," *Neurocomputing*, vol. 312, pp. 324–335, Oct. 2018.

[29] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[30] X. Lin, Y. Duan, Q. Dong, J. Lu, and J. Zhou, "Deep variational metric learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 689–704.

[31] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, no. 2, pp. 1–38, 2009.

[32] O. Rippel, M. Paluri, P. Dollár, and L. Bourdev, "Metric learning with adaptive density discrimination," 2015, *arXiv:1511.05939*. [Online]. Available: http://arxiv.org/abs/1511.05939

[33] K. V. Mardia and P. E. Jupp, *Directional Statistics*, vol. 494. Hoboken, NJ, USA: Wiley, 2009.

[34] J. Xiao, Y. Xie, T. Tillo, K. Huang, Y. Wei, and J. Feng, "IAN: The individual aggregation network for person search," *Pattern Recognit.*, vol. 87, pp. 332–340, Mar. 2019.

[35] H.-M. Yang, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Robust classification with convolutional prototype learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3474–3482.

[36] H.-M. Yang, X.-Y. Zhang, F. Yin, Q. Yang, and C.-L. Liu, "Convolutional prototype network for open set recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Dec. 16, 2020, doi: 10.1109/TPAMI.2020.3045079.

[37] X.-Y. Zhang, C.-L. Liu, and C. Y. Suen, "Towards robust pattern recognition: A review," *Proc. IEEE*, vol. 108, no. 6, pp. 894–922, Jun. 2020.

[38] R. Ranjan, C. D. Castillo, and R. Chellappa, "L_2-constrained softmax loss for discriminative face verification," 2017, *arXiv:1703.09507*. [Online]. Available: http://arxiv.org/abs/1703.09507

[39] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 507–516.

[40] Y. Duan, J. Lu, and J. Zhou, "UniformFace: Learning deep equidistributed representation for face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3415–3424.

[41] K. Zhao, J. Xu, and M.-M. Cheng, "RegularFace: Deep face recognition via exclusive regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1136–1144.

[42] Y. Zheng, D. K. Pal, and M. Savvides, "Ring loss: Convex feature normalization for face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5089–5097.

[43] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.

[44] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[45] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[46] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. NIPS Workshop Autodiff*, 2017.

[47] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 582–591.

[48] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[49] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," 2014, *arXiv:1411.7923*. [Online]. Available: http://arxiv.org/abs/1411.7923

[50] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5353–5360.