Prototype Softmax Cross Entropy: A New Perspective on Softmax Cross Entropy

Qendrim Bytyqi[1], Nicola Wolpert[1], Elmar Schömer[2], and Ulrich Schwanecke[3]

[1] Stuttgart University of Applied Sciences, Germany
{qendrim.bytyqi,nicola.wolpert}@hft-stuttgart.de
[2] Johannes Gutenberg - University Mainz, Germany
schoemer@informatik.uni-mainz.de
[3] RheinMain University of Applied Sciences, Germany
ulrich.schwanecke@hs-rm.de

Abstract. In this work, we consider supervised learning for image classification. Inspired by recent results in the field of supervised contrastive learning, we focus on the loss function for the feature encoder. We show that Softmax Cross Entropy (SCE) can be interpreted as a special kind of loss function in contrastive learning with prototypes. This insight provides a completely new perspective on cross entropy, allowing the derivation of a new generalized loss function, called *Prototype Softmax Cross Entropy* (PSCE), for use in supervised contrastive learning.
We prove both mathematically and experimentally that PSCE is superior to other loss functions in supervised contrastive learning. It only uses fixed prototypes, so no self-organizing part of contrastive learning is required, eliminating the memory bottleneck of previous solutions in supervised contrastive learning. PSCE can also be used equally successfully for both balanced and unbalanced data.

Keywords: Deep learning · Loss function · Contrastive learning · Representation learning · Image classification.

1 Introduction

In supervised learning, neural networks for image classification usually consist of a feature encoder and a classifier. In a first step, the feature encoder converts an input into a feature vector. The feature vector is then transformed into a categorical distribution by the classifier, from which the class assignment can be derived. For balanced data, for which all classes are equally represented in training, the softmax cross entropy (SCE) is often used as a loss function for training the entire network as a whole. We call this the *standard setup* for SCE. This standard setup has the advantage that training can be performed with moderate batch sizes and a relatively small number of epochs, which allows training on affordable hardware.

Inspired by recent findings in the field of supervised contrastive learning (CL) that separating the training with different settings for the feature encoder and the classifier is more efficient [20], we focus in this paper on the loss function for the feature encoder. The training of the feature encoder is done with a non-linear projection network and the training of the classifier with a linear MLP. Furthermore, a multiview batch is used by generating two augmented variants for each object for the feature encoder training. This is necessary for CL methods because there must be at least one positive example per object for the loss functions. We call this setup the *CL setup*.

Khosla et al. [20] have successfully transferred the ideas of self-supervised representation learning to supervised contrastive learning focussing on balanced data. Wang et al. [32] and subsequent Li et al. [25] follow the approach of explicitly defining a prototype for each class towards which the objects of the same class are pulled and from which the objects of the other classes are pushed away. These approaches focus on unbalanced data. It has been shown (see e.g. [20, 32, 25]) that CL increases class separability and hence classification accuracy in supervised learning compared to the standard setup using SCE. The big disadvantage of the state-of-the-arts solutions in supervised CL for balanced [20] and unbalanced [25] data is that the self-organization of the data requires large batch-sizes and many training epochs. A detailed discussion ot the loss functions used in [20, 32, 25] is given in Section 3.

In this work, we propose a new type of loss function for supervised learning. To do so, we first show that SCE has a close similarity to the CL loss function in [32]. Thereby, SCE implicitly uses the standard unit vectors in $\mathbb{R}^C$ as prototypes (in the sense of prototype CL) where C is the number of classes to be determined. Our new perspective on SCE enables two important new ways to improve supervised CL. First, the prototypes in SCE can be chosen to make the best use of the available space on the unit hypersphere. Second, SCE can be embedded into the CL setup in order to take advantage of supervised CL.

We derive a new generalized loss function, which we call *Prototype Softmax Cross Entropy* (PSCE), in which the prototypes can be chosen arbitrarily and for which SCE is a special case. To make optimal use of the available space, we use the corners of a $(C-1)$-simplex in $\mathbb{R}^D$, $D \geq C-1$, as our prototypes. We further embed PSCE into the CL setup. We discuss the advantages of PSCE in the CL setup, both in comparison to SCE in the standard setup and in comparison to the loss functions previously used in supervised CL [20, 32, 25].

Since our solution uses only fixed prototypes, no self-organizing part of CL is required, solving the memory bottleneck of previous solutions. Moreover, PSCE can be used for both balanced and unbalanced data in the same way, achieving state-of-the-art results. In Section 5 we experimentally verify our theoretical considerations with the balanced data sets Cifar10, Cifar100 [21] and ImageNet-1K [28]. The experiments show that the use of PSCE improves the performance of image classification in supervised learning compared to previous methods. For the long-tailed setting, we use Cifar10-LT and Cifar100-LT and PSCE reaches state-of-the-art performance.

2 Previous Work

Variants of SCE. SCE is the most widely used loss function for supervised image classification on balanced data. Some works discuss the weaknesses of SCE and adapt SCE, for example, to reduce sensitivity to noisy labels [30] or to improve the separation of feature vectors of the feature encoder in the case of unbalanced data [6, 12]. For unbalanced data, the output of the classifier trained by SCE is usually biased. Some approaches, such as [17, 27], aim to eliminate this bias by applying a logit adjustment based on the label frequencies. Others distribute sample selection evenly among classes during training [4, 1, 8] or balance the loss function so that low-frequency classes contribute more than high-frequency classes [5, 6, 11]. For both balanced and unbalanced data, it is useful to separate the training of the feature encoder and classifier with different loss functions [20, 19, 36]. For unbalanced data, [19] found that different sampling methods for the training of the feature encoder and the classifier are beneficial.

Contrastive Learning. CL has achieved great success in self-supervised representation learning [3, 13, 7, 35, 9, 15, 16, 29, 31, 34, 14, 10]. The task is to generate semantically meaningful feature vectors (embeddings) generated by a feature encoder for the individual objects of an unlabeled data set. The CL loss is usually not applied directly to the embeddings of the feature encoder, but to embeddings generated by a projection network. This is achieved by contracting the embeddings of augmented variants of an object and pushing them away from the embeddings of other objects [9, 14]. Supervised contrastive learning (SCL) [20] transfers the idea of self-supervised CL to the supervised setting. The key idea is to select positive samples from the same class and negative samples from the other classes. SCL focussed on balanced data on which it shows a very good performance.

Prototype CL. Prototype CL defines one prototype per class and applies the idea of CL such that embeddings of input objects are pulled towards their respective prototypes and pushed away from the prototypes of other classes. In self-supervised CL, this idea has been successfully pursued by determining prototypes using mean vectors of clusters [24] or by using the mean vectors of the available embeddings [23] in few-shot learning. Prototype supervised contrastive learning (PSC) [32] applies this idea to supervised learning by determining prototypes as mean embeddings for each class. To improve robustness to data imbalances, [18] introduce k-positive contrastive learning (KCL), so that the classes have the same number of positive pairs in a batch. Targeted supervised contrastive learning (TSC) [25] builds on this idea and additionally ensures that the mean embeddings of the classes are equally distributed in the feature space even for unbalanced data. This is achieved by selecting prototypes equally distributed on the unit hypersphere.

3 Related Methods

In this section, we shortly discuss the methods respectively loss functions that are most related to our work and introduce our notation.

3.1 Supervised Contrastive Learning (SCL)

Khosla et al. [20] have shown that CL can successfully be transferred from the self-supervised to the supervised setting. Since the data set is labeled, one can form several positive pairs per object. They apply the CL setup with multiview batch and separate training for feature encoder and classifier. The loss function used in [20] is

$$L_{SCL} = -\sum_{\mathbf{y}\in B_f} \frac{1}{|P_{\mathbf{y}}|} \sum_{\mathbf{y}^+ \in P_{\mathbf{y}}} \log \frac{\exp\left(\frac{\mathbf{y}}{||\mathbf{y}||} \cdot \frac{\mathbf{y}^+}{||\mathbf{y}^+||}/\tau\right)}{\sum_{\mathbf{y}_j \in B_f \setminus \{y\}} \exp\left(\frac{\mathbf{y}}{||\mathbf{y}||} \cdot \frac{\mathbf{y}_j}{||\mathbf{y}_j||}/\tau\right)}, \tag{1}$$

where $B_f = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ is the set of embeddings of the current batch, generated by a feature encoder followed by a projection network f, $\tau \in \mathbb{R}^+$ is a scalar temperature hyperparameter and $P_{\mathbf{y}}$ is the set of all positive examples $\mathbf{y}^+$ of $\mathbf{y} \in B_f$. It is a direct adaptation of the loss functions used in the self-supervised case [9, 14]. The major drawback of SCL is its high memory consumption, since a large batch size relative to the number of classes is required to achieve good results (see [32]).

3.2 Prototype Supervised Contrastive Learning (PSC)

Wang et al. [32] have transferred the idea of prototypes to the field of supervised contrastive learning. The prototypes $\mathbf{c}_i$ are normalized mean embeddings of objects belonging to class i. Their loss function,

$$L_{PSC} = -\log \frac{\exp\left(\frac{\mathbf{y}}{||\mathbf{y}||} \cdot \mathbf{c}_j/\tau\right)}{\sum_{k\neq j} \exp\left(\frac{\mathbf{y}}{||\mathbf{y}||} \cdot \mathbf{c}_k/\tau\right)} \tag{2}$$

is also a direct variant of the loss functions in [9, 14], where the number of negative examples is exactly $C - 1$. In contrast to (1), the first sum over the elements in the batch B_f is omitted, since the positive and negative examples no longer depend on the elements in the batch. However, Li et al. [25] have shown that the direct adaption of loss functions from unsupervised CL leads to an unevenly distributed feature space for unbalanced data, as the strong classes tend to capture the feature space.

3.3 Targeted Supervised Contrastive Learning (TSC)

Since loss functions from supervised CL, such as PSC, tend to have strong classes dominating the feature space, Li et al. [25] introduce the targeted supervised contrastive (TSC) loss function

$$
\begin{aligned}
L_{TSC} = \frac{-1}{N(k+1)} \sum_{i=1}^{N} \sum_{y^+ \in B^+_{i,k}} \Bigg(& \log \frac{\exp\left(\frac{\mathbf{y}_i}{||\mathbf{y}_i||} \cdot \frac{\mathbf{y}^+}{||\mathbf{y}^+||} / \tau\right)}{\sum\limits_{\mathbf{y}^- \in B_f} \exp\left(\frac{\mathbf{y}_i}{||\mathbf{y}_i||} \cdot \frac{\mathbf{y}^-}{||\mathbf{y}^-||} / \tau\right)} \\
& + \lambda \cdot \log \frac{\exp\left(\frac{\mathbf{y}_i}{||\mathbf{y}_i||} \cdot \frac{\mathbf{t}_{y_i}}{||\mathbf{t}_{y_i}||} / \tau\right)}{\sum\limits_{\mathbf{y}^- \in B_f} \exp\left(\frac{\mathbf{y}_i}{||\mathbf{y}_i||} \cdot \frac{\mathbf{y}^-}{||\mathbf{y}^-||} / \tau\right)} \Bigg),
\end{aligned}
\tag{3}
$$

which is composed of two parts. The first component is the KCL loss used in [18] which is similar to SCL for balanced data. The second component of TSC attempts to ensure that all classes are evenly distributed in the embedding space when the data is unbalanced. This is achieved by using uniformly distributed prototypes $\mathbf{t}_{y_i}$. This second term in principle follows the idea of PSC, but with different prototypes. As in SCL, the memory requirement of TSC is large.

3.4 Softmax Cross Entropy (SCE)

Softmax cross entropy (SCE) was designed for classification tasks [2, 22]. The aim is to optimize a categorical distribution for an object in such a way that the value of the corresponding class is maximized. In the standard setup, the canonical unit vectors serve as the optimal categorical distribution for the training of the entire network as a whole. If $C \in \mathbb{N}$ is the number of classes, $\mathbf{y} = (y_1, \ldots, y_C)$ is the output of a neural network for an input $\mathbf{x}$ and $\mathbf{t} = (t_1, \ldots, t_C)$ is the canonical unit vector in $\mathbb{R}^C$ which represents the class of $\mathbf{x}$, the SCE loss is defined as

$$
L_{SCE} = -\sum_{i=1}^{C} t_i \cdot \log \frac{\exp(y_i)}{\sum_{k=1}^{C} \exp(y_k)}. \tag{4}
$$

In this standard setup, SCE has the disadvantage that its performance is not optimal and it is useful only for balanced data.

4 Proposed Method

In this section we describe our method and derive our new loss function *Prototype Softmax Cross Entropy* (PSCE). In a first step, we show that SCE has a strong resemblance to PSC.

4.1 Softmax Cross Entropy from the Perspective of CL

Usually SCE is used in the standard setting as a loss function for training the entire network consisting of feature encoder and classifier as a whole. However, as mentioned before, [20] has shown that separating the training from feature encoder and classifier is more efficient in the context of supervised CL for image classification. Based on this, we analyze SCE as a loss function for the feature encoder.

With $\mathbf{y}$ being in class j and $\mathbf{t} = (t_1, \ldots, t_C) = \mathbf{e}_j$, the SCE loss can be rewritten as

$$L_{SCE} = -\sum_{i=1}^{C} t_i \cdot \log \frac{\exp(y_i)}{\sum_{k=1}^{C} \exp(y_k)} \quad = \quad -\log \frac{\exp(y_j)}{\sum_{k=1}^{C} \exp(y_k)} \tag{5}$$

$$= -\log \frac{\exp(\mathbf{y} \cdot \mathbf{e}_j)}{\sum_{k=1}^{C} \exp(\mathbf{y} \cdot \mathbf{e}_k)}. \tag{6}$$

Comparing (6) with the prototypical contrastive loss function (2) shows that SCE has a very similar structure. SCE implicitly uses the canonical unit vectors in $\mathbb{R}^C$ as prototypes in the sense of prototype CL. This new perspective on SCE opens two important new ways to improve supervised CL:

1) The prototypes in SCE can be chosen such that they optimally use the available space on the unit hypersphere.
2) SCE can be embedded into the CL setup to take advantage of the benefits of supervised CL.

In the following, we discuss a direct generalization of SCE for the CL setup.

4.2 Prototype Softmax Cross Entropy (PSCE)

An obvious generalization of softmax cross entropy is to replace the prototypes $\mathbf{e}_i \in \mathbb{R}^C$ with arbitrary pairwise different normalized prototypes $\mathbf{p}_i \in \mathbb{R}^D$, $i = 1, \ldots, C$, for an arbitrary dimension D. We call the resulting loss function

$$L_{PSCE} = -\log \frac{\exp(\mathbf{y} \cdot \mathbf{p}_j)}{\sum_{k=1}^{C} \exp(\mathbf{y} \cdot \mathbf{p}_k)} \tag{7}$$

Prototype Softmax Cross Entropy (PSCE). When comparing PSCE to the CL loss functions SCL (1), PSC (2) and TSC (3) there are two main differences:

1) The output $\mathbf{y}$ is not normalized in PSCE. A side effect of this change is that in PSCE the temperature τ is not necessary.
2) The denominator in PSCE has C summands and not $C - 1$ as in PSC. In PSCE additionally the prototype of the class to which $\mathbf{y}$ belongs is used in the denominator.

We will discuss the effects of these two changes in the following two sections.

4.3 Omitting Normalization

When looking at the three loss functions SCL (1), PSC (2) and TSC (3), it is noticeable that all three use normalized vectors. This is important in CL, as shown experimentally in [33] and analytically in [20]. To see how the lack of normalization in PSCE affects the behavior of the embeddings that pull toward or push away from the normalized prototypes, we reformulate equation (7) (for details see the Supplementary) to

$$L_{PSCE} = \log\left(1 + \sum_{k \neq j} \exp\left(||\mathbf{y}||\Big(\underbrace{\frac{\mathbf{y}}{||\mathbf{y}||} \cdot \mathbf{p}_k}_{=:A_k} - \underbrace{\frac{\mathbf{y}}{||\mathbf{y}||} \cdot \mathbf{p}_j}_{=:B}\Big)\right)\right). \tag{8}$$

For nomalized embeddings and prototypes, A_k and B are cosine similarities with values in $[-1, 1]$. To minimize PSCE, $C_k := A_k - B$ must be minimized. This is achieved if $B = 1$ and the A_k for all $k \neq j$ are as small as possible. Since the prototypes are pairwise distinct and $B = 1$, $A_k < 1$ holds for all $k \neq j$. Therefore, C_k is negative for all $k \neq j$ and the expression in (8) becomes minimal when $||\mathbf{y}||$ becomes maximal.

This leads to the following differences in the behavior of the embeddings during training between SCL, PSC and PSCE (see also Figure 1), while TSC is a combination of SCL and PSC.

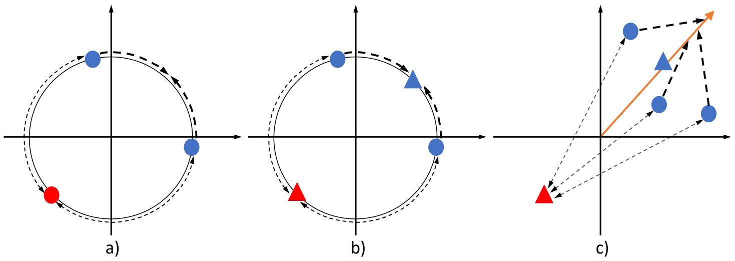

Fig. 1. Behavior of 2D embeddings during training using a) SCL, b) PSC and c) PSCE.

a) In SCL (see equation (1) and Figure 1 a)), the embeddings are normalized and thus all lie on the unit sphere. The embeddings of the positive pairs (blue dots) are contracted and pushed away from the negative examples (red dots).
b) In PSC (see equation (2) and Figure 1 b)), the embeddings are also normalized. Here, the embeddings of one class are pulled towards a prototype (blue triangle) and pushed away from the prototypes of the other classes (red triangle). Again, all embeddings and prototypes lie on the unit sphere.

c) In contrast, in PSCE (see equation (7) and Figure 1 c)) the embeddings $\mathbf{y}$ are not pulled towards the corresponding prototypes on the unit sphere, but are pulled outwards in the direction of its orange corresponding prototype vector. What remains the same is that the embeddings $\mathbf{y}$ are pushed away from the prototypes of the other classes. This gives the embeddings of different classes the possibility to take a further distance from each other.

A visualization of the learned embeddings of the datasets Cifar10 (balanced) and Cifar10-LT (unbalanced), for $D = 2$, is given in Figure 2. It can be seen that all classes are equally well separated in both scenarios.

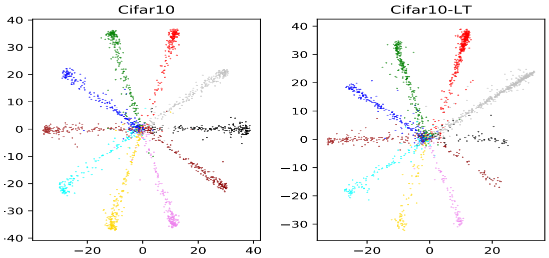

Fig. 2. Visualization of the learned embeddings on Cifar10 and Cifar10-LT.

4.4 Adding Hard Mining Support

Our loss function PSCE differs from PSC (2) in that the denominator also contains the summand $k = j$ for the corresponding prototype $\mathbf{p}_j$. The importance of this summand is shown analytically and also experimentally in the following.

As shown in the Supplementary, the gradient of the partial derivative of PSCE with respect to $\mathbf{y}$ is:

$$\frac{\partial L_{PSCE}}{\partial \mathbf{y}} = -\frac{\sum_{j\neq k}(\mathbf{p}_j - \mathbf{p}_k)\cdot \exp(\mathbf{y}\cdot \mathbf{p}_k)}{\sum_{j\neq k}\exp(\mathbf{y}\cdot \mathbf{p}_k) + \exp(\underbrace{||\mathbf{y}||\cdot \frac{\mathbf{y}}{||\mathbf{y}||}\cdot \mathbf{p}_j)}_{=:A}}. \tag{9}$$

One can see that in (9) the derivative of PSCE differs from the one of PSC by the additional term $\exp(A)$ in the denominator. The larger A is, the larger the denominator and consequently the smaller the gradient and the smaller the contribution of $\mathbf{y}$ to the loss function. This supports hard mining, because in this case the characteristic vector $\mathbf{y}$ is already far towards its prototype and thus well separated from other classes.

For PSCE the additional term for $k = j$ in the denominator is essential for the training. If it is removed, the loss explodes during minimization. This is clear, since in this case equation (8) becomes (for details see the Supplementary)

$$L = \log\Big(\sum_{k \neq j} \exp\Big(||\mathbf{y}||\underbrace{\Big(\frac{\mathbf{y}}{||\mathbf{y}||} \cdot \mathbf{p}_k - \frac{\mathbf{y}}{||\mathbf{y}||} \cdot \mathbf{p}_j\Big)}_{=:C_k}\Big)\Big). \tag{10}$$

As for the PSCE loss (see (8)), L becomes minimal when $||\mathbf{y}||$ becomes maximal and $C_k < 0$. Since $\mathbf{y}$ is unbounded, the argument of the logarithm tends to 0. Consequently, L goes to negative infinity during optimization. In PSCE loss, however, the argument of the logarithm tends to 1 due to the additional summand in the denominator. Therefore, PSCE is lower bounded by 0. PSC is able to work without this additional term, because $\mathbf{y}$ is normalized and therefore the loss function is lower bounded. We show this in the Supplementary.

4.5 Choice of Prototypes

In this subsection, we present two methods for selecting the prototypes in PSCE in a meaningful way if no semantic information about the classes is available. Already TSC [25] has shown that it is useful to choose the prototypes uniformly distributed on a D-dimensional unit hypersphere. Our experiments will show that it is very important that the prototypes have pairwise equal distance and therefore build the vertices of a regular simplex. We prove in the Supplementary that this is only possible for $D \geq C - 1$, with C beeing the number of classes. In TSC, this condition is not always fulfilled, because a fixed dimension D is chosen. In our experiments, we investigate the effect of different values for D.

First, as in the original SCE but within the CL setup, we choose the prototypes as the canonical unit vectors in $\mathbb{R}^D$ with $D = C$. We call this PSCE$_c$, the canonical version of PSCE. In PSCE$_c$ the prototypes have the pairwise Euclidean distance $\sqrt{2}$ from each other.

Second, we place the C prototypes on the vertices of a regular $(C-1)$-simplex on the unit hypersphere in $\mathbb{R}^D$ with $D = C$ and the origin as the center point. Here the pairwise Euclidean distance between the prototypes is $\sqrt{2C/(C-1)} > \sqrt{2}$ which is optimal. For a proof consider the Supplementary. We call this PSCE$_s$, the simplex version of PSCE. It is easy to see that the pairwise distance of the prototypes in PSCE$_s$ converges against $\sqrt{2}$ (as in PSCE$_c$) if the number of classes C grows.

5 Experiments

In this section, we present our experiments, details of our implementation setup, which is the CL setup and comparable to [20] for balanced and to [32, 25] for unbalanced data, as well as information about the data sets used can be found in the Supplementary. We also test SCE in the standard setup. For this, we use the same implementation as for the CL methods, but only with singleview batch (only one augmentation per object) and one stage training.

5.1 Image Classification on Balanced Data

In this section, we compare our loss functions PSCE$_c$ and PSCE$_s$ with SCE and with SCL for balanced settings. We used the data sets Cifar10, Cifar100 [21] and ImageNet-1K [28].

In Table 1 we compare the accuracy (%) of PSCE$_c$ and PSCE$_s$ with the ones of SCE and SCL on the balanced data sets CIFAR-10 and CIFAR-100. To be comparable, we use a batch size of 32 for all three methods and train with 300 epochs. In addition, we cite the original result from [20] with batch size 1024 and 1000 epochs. SCL needs a large batch size to achieve good results because the positive and negative pairs are taken out of the batch. The table shows that PSCE$_c$ and PSCE$_s$ with equal batch size and equal number of epochs clearly beat the other methods. They are even better than the original, very computationally intensive setup of SCL, for Cifar100 even significantly better.

	SCE (BS 32)	SCL (BS 32)	SCL (BS 1024)	PSCE$_c$ (BS 32)	PSCE$_s$ (BS 32)
Cifar10	93.8	95.4	96.0 [20]	96.0	**96.2**
Cifar100	74.4	76.1	76.5 [20]	**78.8**	**78.8**

Table 1. Top 1 accuracy comparison on Cifar10 and Cifar100 with ResNet-50.

Besides the evaluation of Top-1 accuracy, we also perform tests with a KNN classifier and Top-5 accuracy with a linear classifier on Cifar100 for SCE, PSCE$_c$ and PSCE$_s$ with batch size 32. In all cases, PSCE$_c$ and PSCE$_s$ outperform SCE. Details can be found in Table 2.

	Top-1 Accuracy Linear	Top-5 Accuracy Linear	KNN
SCE	74.4	92.9	74.9
PSCE$_c$	**78.8**	94.7	78.7
PSCE$_s$	**78.8**	**94.9**	**78.8**

Table 2. Top 1 and Top 5 accuracy with linear evaluation and Top 1 accuracy with KNN classifier on Cifar100 with ResNet-50.

On the much larger data set ImageNet-1K, we again compare the accuracy of PSCE$_c$ and PSCE$_s$ with SCE and SCL in Table 3. For resource reasons, we tested PSCE$_c$ and PSCE$_s$ with a singleview batch size of 416 and 120 epochs pretraining and to ensure a fair comparison, we quote the result for SCE and SCL presented in Figure 4 in [20] with a batch size 512 and 700 epochs pretraining. Again, PSCE$_c$ and PSCE$_s$ perform better than the other methods.

	SCE (700 E)	SCL (700 E)	PSCE$_c$ (120 E)	PSCE$_s$ (120 E)
ImageNet-1K	$<$ 75.0 [20]	$<$ 76.0 [20]	76.6	**76.7**

Table 3. Comparison on ImageNet-1K with ResNet-50.

5.2 Image Classification on Unbalanced Data

For comparison on the long-tailed data, we use Cifar10-LT and Cifar100-LT. We sample CIFAR10-LT and CIFAR100-LT with an exponential decay with different imbalance ratios ρ. Hereby ρ is the cardinality of the most represented divided by the least represented class. Table 4 shows that PSCE$_c$ and PSCE$_s$ are capable of producing state-of-the-art results even for unbalanced data.

Data set	Cifar10-LT			Cifar100-LT		
Imbalance Ratio ρ	100	50	10	100	50	10
SCE [25]	70.4	74.8	86.4	38.3	43.9	55.7
BSCE-SCE [11]	72.4	78.1	86.8	38.6	44.6	57.1
Focal [26]	70.4	76.7	86.7	38.4	44.3	55.8
BSCE-Focal [11]	74.6	79.3	87.1	39.6	45.2	58.0
LDAM-DRW [6]	77.0	80.9	88.2	42.0	46.2	58.7
KCL [18]	77.6	81.7	88.0	42.8	46.3	57.6
TSC [25]	79.7	82.9	88.7	43.8	47.4	59.0
PSC [32]	78.8	83.9	**90.1**	**45.0**	48.9	**62.4**
PSCE$_c$	81.5	**84.4**	89.1	44.6	**49.0**	59.5
PSCE$_s$	**81.6**	84.3	89.5	44.7	48.9	59.0

Table 4. Top 1 accuracy for different imbalance ratios ρ with ResNet-32.

5.3 Ablation on Cifar10 and Cifar100

Batch Size Since the CL setup uses a multiview batch, its batch size is always twice the size of the SCE standard setup. To exclude that this imbalance has an effect, we also compare PSCE$_c$ and PSCE$_s$ with SCE, where we double the batch size for the latter, see Figure 3. It can be seen that PSCE$_c$ and PSCE$_s$ still achieve better accuracy than SCE overall.

Effect of the CL Setup In the next experiment, we investigate the difference in accuracy between SCE and PSCE$_c$. Both use the canonical unit vectors. Therefore, we examine the individual component effects of the CL setup, namely the multiview batch and the two stage training. We use SCE as a baseline. SCE has a singleview batch and a one stage training (i.e. 1 view and 1 stage: 1v1s). First, we examine the effect of the multiview batch, i.e. two views but still a one stage training (SCE2v1s). This can increase the accuracy by 1.6% for Cifar10 and by 3.2% for Cifar100. Similar improvement is achieved by the two stage

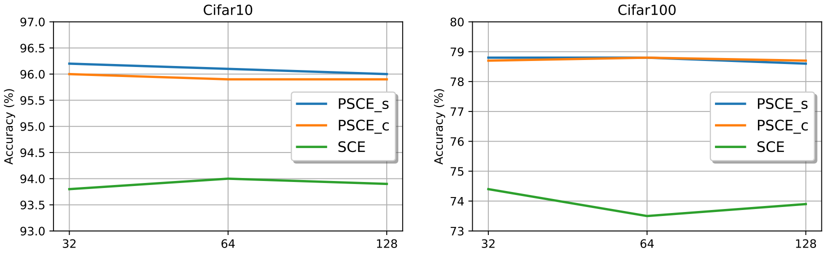

Fig. 3. Comparison of different batch sizes on Cifar10 and Cifar100.

training, but a singleview (SCE1v2s). Here, the accuracy can be increased by 1.5% for Cifar10 and 3.2% for Cifar100. The greatest benefit is achieved by the combination of both components (PSCE$_c$), which results in an improvement of 2.2% for Cifar10 and 4.4% for Cifar100 compared to SCE.

	SCE	SCE2v1s	SCE1v2s	PSCE$_c$
Cifar10	93.8	95.4	95.3	**96.0**
Cifar100	74.4	77.6	77.6	**78.8**

Table 5. Effect of multiview batch and two stage training in the CL setup.

Normalization and Additional Summand in the Denominator In this experiment we confirm our theoretical considerations from Section 4 (see Table 6). For the experiments we use PSCE$_s$ on Cifar10 and Cifar100, the results for PSCE$_c$ were comparable. Our first consideration was that PSCE$_s$ works without normalization (discussed in 4.3). That this is useful is confirmed in our experiments because, as $\mathbf{y}$ is normalized (notation: w/ N), the accuracy of PSCE$_s$ decreases. Second, as in PSC, we normalize $\mathbf{y}$ (w/ N) but remove the additional summand (notation: w/o S) as discussed in 4.4. In this case, the accuracy also drops. We conclude that in the case of fixed prototypes, it is advantageous if the embeddings are not normalized. Furthermore, the denominator should contain all summands with the respective prototypes, which reinforces our analytical result.

	PSCE$_s$	PSCE$_s$ w/ N	PSCE$_s$ w/ N w/o S
Cifar10	**96.2**	95.7	95.0
Cifar100	**78.8**	77.9	77.6

Table 6. Effect of normalized $\mathbf{y}$ and the additional summand in the denominator.

Selection of Prototypes and Dimension D Next, we experimentally support our hypothesis that maximum pairwise equal distance of prototypes on the unit hypersphere is important. For Cifar10 and Cifar100, we compare the prototypes of PSCE$_s$ which have maximum possible pairwise distance with the canonical unit vectors of PSCE$_c$ and with randomly selected prototypes. Again, we use $D = 100$ for Cifar100 and $D = 10$ for Cifar10. Table 7 shows that random prototypes perform worst. The best way is to choose prototypes with pairwise equal and maximum distance.

	Rand.	PSCE$_c$	PSCE$_s$
Cifar10	95.4	96.0	**96.2**
Cifar100	78.1	**78.8**	**78.8**

Table 7. Comparison of different choices of prototypes: random, canonical unit vectors and corners of an optimal regular simplex.

The distance between the prototypes for PSCE$_c$ is always $\sqrt{2}$. For PSCE$_s$ the distance between the prototypes for Cifar100 is $\sqrt{200/99} = \sqrt{2.\overline{02}}$ and for Cifar 10 it is $\sqrt{20/9} = \sqrt{2.\overline{2}}$. For Cifar100, the two distances of PSCE$_s$ and PSCE$_c$ are almost the same and one can see that also the accuracies are identical. For Cifar10, on the other hand, the difference between the distances is noticeable and this is also reflected in the accuracies.

Cifar10

D	2	5	10	20
Acc.	94.7	95.4	**96.2**	96.1
Diff. Dist.	1.38	0.32	0	0

Cifar100

D	2	50	100	200
Acc.	58.1	77.5	**78.8**	**78.8**
Diff. Dist.	1.94	0.33	0	0

Table 8. Effect of different dimensions D on the Cifar10 an Cifar100 data sets.

We then study the effect of different dimensions D on Cifar10 and Cifar100 in Table 8. We use PSCE$_s$ for $D \geq C - 1$. As mentioned, for $D < C - 1$ there exists no regular $(C-1)$-simplex in $\mathbb{R}^D$. For these cases, we use a particle system to select the points at maximum distance from each other.

We additionally specify in the table for each value of D the difference between the largest and the smallest distance between two prototypes. This is a good measure, to what extent a regular simplex with pairwise equal distance is reached. If the value is zero, all prototypes have the same pairwise distance and lie on the corners of a regular simplex. The larger the value, the less the simplex is fulfilled. The result of our experiment reinforces our assumption that pairwise equal distance between prototypes is important: the closer the prototypes come to a regular simplex, the higher the accuracy. Furthermore, one can see that once a simplex is possible, increasing the dimension no longer increases the accuracy.

6 Conclusion and Future Work

In this work, we derive the statement that Softmax Cross Entropy (SCE) is strongly associated with prototype contrastive learning (CL), where the canonical unit vectors serve as prototypes. We present Prototype Softmax Cross Entropy (PSCE), a generalization of SCE allowing arbitrary prototypes. We consider the case where no information is available about the relation between the classes and thus select prototypes with pairwise equal distance on a regular simplex. We present two versions of PSCE, namely PSCE$_c$ and PSCE$_s$, which produce state-of-the-art results for both balanced and unbalanced data. Since our solution uses fixed prototypes, no self-organizing part from CL is needed. Our experiments show that PSCE allows smaller batch sizes and less training epochs to achieve state-of-the-art results.

In the future, we want to investigate how semantic information about different classes can be used to place prototypes more specifically on the unit hypersphere.

7 Acknowledgments

This work was partially supported by the “Research at Universities of Applied Sciences” program of the German Federal Ministry of Education and Research, funding code 13FH010IX6.

References

1. Ando, S., Huang, C.Y.: Deep over-sampling framework for classifying imbalanced data. Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2017)
2. B., B.E., Frank, W.: Supervised learning of probability distributions by neural networks. Advances in Neural Information Processing Systems (1988)
3. Bardes, A., Ponce, J., LeCun, Y.: Vicreg: Variance-invariance-covariance regularization for self-supervised learning. International Conference on Learning Representations (2022)
4. Buda, M., Maki, A., Mazurowski, M.A.: A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks (2018)
5. Byrd, J., Lipton, Z.: What is the effect of importance weighting in deep learning? International Conference on Machine Learning (2019)
6. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. Advances in Neural Information Processing Systems (2019)
7. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. Neural Information Processing Systems (2020)
8. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research (2002)
9. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020)
10. Chen, X., He, K.: Exploring simple siamese representation learning. Computer Vision and Pattern Recognition (2021)
11. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. Computer Vision and Pattern Recognition (2019)
12. Elsayed, G., Krishnan, D., Mobahi, H., Regan, K., Bengio, S.: Large margin deep networks for classification. Advances in neural information processing systems (2018)
13. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Carl Doersch, B.A.P., Guo, Z.D., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M.: Bootstrap your own latent a new approach to self-supervised learning. Neural Information Processing Systems (2020)
14. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
15. Hénaff, O.J., Srinivas, A., Fauw, J.D., Razavi, A., Doersch, C., Eslami, S.M.A., van den Oord, A.: Data-efficient image recognition with contrastive predictive coding. arXiv preprint arXiv:1905.09272 (2019)
16. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. International Conference on Learning Representations (2019)
17. Hong, Y., Han, S., Choi, K., Seo, S., Kim, B., Chang, B.: Disentangling label distribution for long-tailed visual recognition. Computer Vision and Pattern Recognition (CVPR) (2021)
18. Kang, B., Li, Y., Xie, S., Yuan, Z., Feng, J.: Exploring balanced feature spaces for representation learning. International Conference on Learning Representations (2020)

19. Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., Kalantidis, Y.: Decoupling representation and classifier for long-tailed recognition. International Conference on Learning Representations (2017)
20. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. Advances in Neural Information Processing Systems (2020)
21. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report (2009)
22. Levin, E., Fleisher, M.: Accelerated learning in layered neural networks. Complex Systems (1988)
23. Li, J., Zhou, P., Xiong, C., Hoi, S.C.: Prototypical networks for few-shot learning. Neural Information Processing System (2017)
24. Li, J., Zhou, P., Xiong, C., Hoi, S.C.: Prototypical contrastive learning of unsupervised representations. International Conference on Learning Representations (2021)
25. Li, T., Cao, P., Yuan Yuan, L.F., Yang, Y., Feris, R.S., Indyk, P., Katabi, D.: Targeted supervised contrastive learning for long-tailed recognition. Conference on Computer Vision and Pattern Recognition (2022)
26. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. International Conference on Computer Cision (2017)
27. Menon, A.K., Jayasumana, S., Rawat, A.S., Jain, H., Veit, A., Kumar, S.: Long-tail learning via logit adjustment. International Conference on Learning Representations (2020)
28. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (2015)
29. Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S.: Time-contrastive networks: Self-supervised learning from video. International Conference on Robotics and Automation (2018)
30. Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., Fergu, R.: Training convolutional networks with noisy labels. arXiv preprint arXiv:1406.2080 (2014)
31. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. arXiv preprint arXiv:1906.05849 (2019)
32. Wang, P., Han, K., Wei, X.S., Zhang, L., Wang, L.: Contrastive learning based hybrid networks for long-tailed image classification. Computer Vision and Pattern Recognition (2021)
33. Wu, Z., Efros, A.A., Yu, S.X.: Improving generalization via scalable neighborhood component analysis. European Conference on Computer Vision (2018)
34. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. Computer Vision and Pattern Recognition (2018)
35. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. Proceedings of Machine Learning Research (2021)
36. Zhou, B., Cui, Q., Wei, X.S., Chen, Z.M.: Bilateral-branch network with cumulative learning for long-tailed visual recognition. Conference on Computer Vision and Pattern Recognition (2020)