Neural Processing Letters (2022) 54:4735–4749
https://doi.org/10.1007/s11063-022-10830-9

COLAM: Co-Learning of Deep Neural Networks and Soft Labels via Alternating Minimization

Xingjian Li[1,2] · Haoyi Xiong[1] · Haozhe An[1] · Chengzhong Xu[2] · Dejing Dou[1]

Accepted: 5 April 2022 / Published online: 13 May 2022

Abstract
Softening labels of training datasets with respect to data representations has been frequently used to improve the training of deep neural networks. While such a practice has been studied as a way to leverage "privileged information" about the distribution of the data, a well-trained learner with soft classification outputs should be first obtained as a prior to generate such privileged information. To solve such a *"chicken-and-egg"* problem, we propose COLAM framework that Co-Learns DNNs and soft labels through Alternating Minimization of two objectives—(a) the training loss subject to soft labels and (b) the objective to learn improved soft labels—in one end-to-end training procedure. We performed extensive experiments to compare our proposed method with a series of baselines. The experiment results show that COLAM achieves improved performance on many tasks with better testing classification accuracy. We also provide both qualitative and quantitative analyses that explain why COLAM works well.

Keywords Deep learning · Neural networks · Soft label

Xingjian Li and Haoyi Xiong contributed equally to this work.

✉ Haoyi Xiong
xionghaoyi@baidu.com

Xingjian Li
lixingjian@baidu.com

Haozhe An
v_anhaozhe@baidu.com

Chengzhong Xu
czxu@um.edu.mo

Dejing Dou
doudejing@baidu.com

1 Big Data Lab, Baidu Research, Beijing, China

2 University of Macau, Zhuhai, China

1 Introduction

Recent years have witnessed rapid developments in deep neural networks [6, 21–23] and their widespread applications in a variety of tasks [5, 12]. Due to the over-parameterized nature of deep neural networks [19, 31], tons of tricks have been invented to improve the generalization performance of deep learning through regularizing the training procedure, such as weigh decay [13, 32], DropOut [9], stochastic pooling [30], data augmentation [14], as well as using perturbed labels [26] or soft labels [23] (drawn from the continuous space) to replace the original hard labels (zero-one coded) of training samples. In this work, we study the problem of learning optimal soft labels for training samples subject to the deep learning process, and further propose novel algorithm that intends to co-learn both "best" soft labels and deep neural networks through an end-to-end training procedure.

Researchers have widely adopted label smoothing to optimize a broad range of tasks, including but not limited to image classification [11, 20, 33], speech recognition [2], and machine translation [24]. The key principle here to regularize the deep learning procedure with certain privileged prior information [15, 26] embedded in the soft labels. With a set of predefined rules, label smoothing [23] was first proposed to soften the hard labels to regularize the training objectives with smoothness. In addition to using predefined mappings, learning from the soft classification outputs (e.g., logits) of a well-trained teacher neural network (often named as knowledge distillation [8]) could also improve the generalization performance significantly. In our research, we soften labels using well-trained neural networks, so as to incorporate the privileged knowledge of teacher network [15]. More specifically, due to the lack of well-trained models in advance, the proposed algorithm is expected to learn optimal soft labels from the DNN outputs during the training process, i.e., under self-distillation settings.

To achieve the goal, we propose a novel deep learning algorithm, namely COLAM—the CO-Learning of deep neural networks and soft labels via Alternating Minimization. During the training procedure, COLAM alternatively minimizes two learning objectives: (i) the training loss subject to the target (soft) labels, and (ii) the loss to learn soft label design subject to the logit outputs of learned labels. Compared to the existing solution that either use the raw soft prediction results as the soft labels for self-distillation, or leverage pre-trained models as teacher networks with additional computation cost required, COLAM uses one end-to-end training procedure to effectively learn both soft labels (of all training samples) and the model all in once. COLAM improves the generalization of deep learning through softening the labels with "privileged" information while enjoying the same computation cost of vanilla training.

The contributions made in this paper are as follows. We study the technical problem of co-learning soft labels and deep neural network during one end-to-end training process in a self-distillation setting. We design two objectives to learn the model and the soft labels respectively, where the two objective functions depend on each other. We further propose COLAM algorithm that achieves the goal through alternatively minimizing two objectives. Extensive experiments have been done using real-world image classification datasets under the both supervised learning and transfer learning settings. We compare COLAM with a bunch of baselines algorithms using soft labels and perturbed labels. The experiment results showed that COLAM can significantly outperform baseline methods with significantly higher classification accuracy (1–2%) using comparable computation cost.

2 Related Work and Backgrounds

Label smoothing (LS) was first introduced in [23] to enhance the performance of Inception model on ImageNet [4]. This traditional label smoothing is a weighted average of ground-truth label. Formally, given a ground-truth label $\boldsymbol{y} = (\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^n)$ in a classification task for n classes, we have $\forall i \in [1, n]$, $\boldsymbol{y}^i \in \{0, 1\}$ and $\sum_i \boldsymbol{y}^i = 1$. If $\boldsymbol{y}_l = 1$, then l is the correct class to which that sample belongs. The softened label is obtained by

$$\boldsymbol{y}^i_{soft} = \boldsymbol{y}^i(1 - \epsilon) + \epsilon/n, \tag{1}$$

where ϵ is a hyperparameter to control the degree of softness, i.e. larger ϵ results in softer labels. Note that the superscripts represent the indices. The hard label is replaced by the softened label when computing the cross entropy loss between label and predicted probabilities. Mathematically, the cross entropy between the ground-truth targets y and a predicted probability distribution p is

$$H(\mathbf{y}, p) = \sum_i -\boldsymbol{y}^i \log(p^i). \tag{2}$$

Now the soft label substitutes the ground-truth hard label in the cross entropy function, giving rise to

$$H(\mathbf{y}, p) = \sum_i -\boldsymbol{y}^i_{soft} \log(p_i). \tag{3}$$

Note that while the primitive form of the softened loss function should be represented by a distribution over all categories [23], the simplified form as in Eq. 1 is usually used in practice as recommended by [23], which uses a constant ϵ. This noisy loss result enables the network to reduce the chances of being overconfident while making predictions, thus regularizing the network. Besides following a uniformed distribution to produce soft labels, label smoothing can be more dynamic. [19] shows that, with a slight modification of the KL Divergence direction, "confidence penalty" regularizer is equivalent to label smoothing. This regularizer encourages predictions to have larger entropy and lower confidence on the most probable class. It is achieved by softening the model output.

Furthermore, adding noise to labels produces similar effects as label smoothing. DisturbLabel [26] is a regularizer in the loss layer of a network. It adds noise to labels during training by randomly changing a correct label $\boldsymbol{y}$ to another one-hot label $\tilde{\boldsymbol{y}}$. This permutation of the elements in labels happens under a certain fixed likelihood. [26] points out DisturbLabel has the same expected gradient as label smoothing because $\mathbb{E}(\tilde{\boldsymbol{y}}) = \boldsymbol{y}_{soft}$. However, DisturbLabel outperforms traditional LS on many datasets, likely because randomness in the algorithm contributes to the success. Although these regularizers bring improvements in generalization, little to no dataset knowledge is involved to produce soft labels.

3 COLAM Algorithm Design

In this section, we first present the overall learning procedure with the design of two objectives.

3.1 Learning Procedure

Let $\mathcal{D} \in \mathcal{X} \times \mathcal{Y}$ be the training set which contains $|\mathcal{D}|$ labelled training samples and C classes. Each sample is denoted by $(\mathbf{x_i}, y_i) \in \mathcal{D}$, where $y_i \in \{1, 2, \ldots, C\}$. We define our

Algorithm 1: COLAM Algorithm

Input: Initial parameters $\boldsymbol{\theta}_{M_0}$; the temperature T; and the set of training samples $\mathcal{D}$
Output: Deep neural network model parameters $\boldsymbol{\theta}_{M_m}$

for $n = 1, 2, 3, \ldots, m$ **do**
  /* Training from $\boldsymbol{\theta}_{M_{n-1}}$ with t epochs */
  if M_n *is the first stage* M_1 **then**
    /* Using Original Hard Labels */
    $\boldsymbol{\theta}_{M_n} \leftarrow \text{argmin}_\theta \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}(\theta; (\boldsymbol{x}_i, y_i), T)$
  else
    /* Using (Updated) Soft Labels */
    $\boldsymbol{\theta}_{M_n} \leftarrow \text{argmin}_\theta \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}(\theta; (\boldsymbol{x}_i, \boldsymbol{y}_{i,\text{soft}}), T)$
  /* Updating the Soft Labels using $\boldsymbol{\theta}_{M_n}$ */
  for $i = 1, 2, 3, \ldots, |\mathcal{D}|$ **do**
    Obtaining peer samples of $(\boldsymbol{x}_i, y_i)$
    $Y_{y_i} \leftarrow \text{getPeerSamples}(\boldsymbol{x}_i, y_i)$;
    $\boldsymbol{y}'_i \leftarrow \text{arg min} \sum_{\forall y \in \mathbb{R}^{|C|} \forall (x,y) \in Y_{y_i}} \text{dist}(y; f(x; \boldsymbol{\theta}_{M_n}))$
    $\boldsymbol{y}_i \leftarrow \text{softmax}(\boldsymbol{y}'_i / T)$
  return $\boldsymbol{\theta}_{M_m}$

objective deep neural network network $\boldsymbol{f}(\mathbf{x}; \boldsymbol{\theta}) : \mathcal{X} \mapsto \mathcal{Y}$ parameterized with $\boldsymbol{\theta}$ as the mapping function.

COLAM splits the overall training procedure into m equal-length stages $M = \{M_1, M_2, \ldots, M_m\}$, with each stage consisting of t epochs. Denoting n as the number of epochs the standard training procedure requires, we select m and t that satisfy $m \times t = n$ in practice. Therefore COLAM does not increase the overall computation time. At the end of every stage, COLAM updates the soft labels based on the current checkpoint. The training procedure differs a bit between the first stage and the remaining as described in the following paragraphs.

In the first stage, COLAM uses the original hard labels of training samples to train deep neural network with t epochs and obtains θ_{M1}. Then, COLAM computes the soft label for every sample (i.e., $\boldsymbol{y}_{i,soft}$ for sample $(\boldsymbol{x}_i, y_i)$) in the training set through minimizing the *Loss of Soft Label Learning*.

From the second stage to the final stage of the training procedure, COLAM continues the deep learning procedure using the training samples with soft labels $((\boldsymbol{x}_i, \boldsymbol{y}_{i,soft}))$ via minimizing the *Loss of Model Learning*, and repeats the soft label computation by the end of stage. Through alternatively minimizing the training loss and soft label design loss, COLAM is expected to reach the convergence of deep learning and outputs the $\boldsymbol{\theta}_{M_m}$ as the results of soft label and model co-learning. The model would be trained using $m \times t$ epochs.

The complete algorithm is illustrated in Algorithm 1. We now introduce in detail the *Loss of Model Learning* in Section 3.2 and the *Loss of Soft Label Learning* in Section 3.3.

3.2 Loss of Deep Neural Network Training

COLAM simply uses the cross-entropy function as the loss to train deep neural networks. For the first stage, COLAM computes the DNN training loss using hard labels, while it starts leveraging the soft labels from the second stage. Given a pair of predictor and label (x, y), the parameter θ and the temperature T for softmax, COLAM considers the cross-entropy

loss as follow.

$$\mathcal{L}(\theta; (x, y), T) = -\sum_{j=1}^{C} y^j \log\left(\text{softmax}^j(f(x;\theta)/T)\right), \tag{4}$$

where y^j refers to the j^{th} dimension of the label y, $\text{sofmax}^j(\cdot)$ is the j^{th} dimension of the input and T controls the softness of the label distribution. We will demonstrate and discuss how T influences the training later in Section 5.3. The softmax function is defined as follow.

$$\text{softmax}^j(y) = \frac{\exp(y^j)}{\sum_{c\in\{1,\ldots C\}\setminus\{j\}} \exp(y^c)}, \ \forall 1 \le j \le C. \tag{5}$$

Note that, for the first stage, COLAM uses $(x, y) \in \mathcal{D}$ referring to the training sample with hard labels. From the second stage, COLAM uses $(x, y) \in \{(\boldsymbol{x}_i, \boldsymbol{y}_{i,\text{soft}})|\forall 1 \le i \le |\mathcal{D}|\}$ referring to the sample with the soft labels. Please refer to lines 3–8 of Algorithm. 1 for details.

3.3 Loss of Soft Label Learning

To achieve the better generalization while lowering the computation complexity, COLAM assumes all samples of the same class share the same soft label.

Peer samples In this way, to learn the soft label, COLAM first retrieves the peer samples for every training sample. Given $\mathcal{D}_i = (\mathbf{x_i}, y_i)$, its peer sample set (denoted as Y_{y^i}) is defined as $Y_{y^i} = \left\{\forall(\mathbf{x}, y) \in \mathcal{D} \mid y = y_i\right\}$.

Soft label loss Given the set of peer samples Y_{y^i} for the class y_i, COLAM computes the soft label $\boldsymbol{y}'_i$ through minimizing the distances between the soft label to the soft prediction results of every sample in Y_{y^i} (please see also in Line 13 of Algorithm. 1). Such that, COLAM simply defines the distance as follow.

$$\text{dist}\left(y; y'\right) = \frac{1}{2}\|y - y'\|_2^2, \tag{6}$$

where y' refers to the soft prediction result and y is the learning objective of soft labels. To simplify the computation, line 13 of Algorithm 1 is equivalent to estimate the mean soft labels among all peer samples. With the $\boldsymbol{y}'_i$ obtained, COLAM uses softmax to further normalize the vector. Finally, COLAM uses $\boldsymbol{y}_i$ to replace y_i as the soft label for further computation. Please refer to lines 9–14 of Algorithm. 1 for details.

3.4 Relationship with State-of-the-Art Algorithms

Compared with widely used label promotion techniques such as label smoothing [19, 23] and label perturbation [26], our algorithm intends to exploit the underlying semantic information of the data distribution, which alternately serves as the prior and the optimization objective. Utilizing such *privileged information* speeds up convergence and facilitates learning more meaningful deep representations, as demonstrated in Section 5.1 and 5.2. Further, our algorithm is as efficient as these competitors because the class-specific soft labels can be progressively updated without additional training.

Another line of related studies involve a well-trained teacher model to generate soft labels for each training sample, which are used as the learning target for the student model [8]. As there may exist inconsistency between the original label and the real object after random

Table 1 Test accuracy on CIFAR10

Model	HL	LS [23]	CP [19]	DL [26]	COLAM
DenseNet-BC-100	94.74±0.12	94.81±0.13	**94.95±0.19**	94.81±0.08	94.84±0.10
WideResNet40x4	95.59±0.08	95.58±0.05	95.51±0.01	95.66±0.08	**95.69±0.07**
PreActResNet34	95.02±0.09	95.01±0.05	95.12±0.09	95.19±0.13	**95.21±0.06**
ResNeXt29_8x64d	94.82±0.11	95.12±0.04	95.79±0.22	95.52±0.23	**95.91±0.17**

HL refers to models trained with standard Stochastic Gradient Descent optimizer using hard labels. LS are models improved by using Label Smoothing technique. CP refers to models regularized by Confidence Penalty. DL refers to DisturbLabel

Table 2 Test accuracy on CIFAR100

Model	HL	LS [23]	CP [19]	DL [26]	COLAM
DenseNet-BC-100	75.15±0.38	75.51±0.21	75.43±0.17	75.91±0.37	**76.15±0.14**
WideResNet40x4	77.99±0.24	78.08±0.10	77.81±0.33	78.45±0.23	**78.68±0.15**
PreActResNet34	77.01±0.32	78.27±0.14	77.55±0.24	77.62±0.31	**78.69±0.23**
ResNeXt29_8x64d	80.17±0.31	81.03±0.17	80.30±0.23	80.64±0.18	**81.47±0.28**

Same abbreviations are used as found in Table 1

cropping, [1] improves [8] by generating more accurate soft labels based on cropped images. Although being more effective, these methods have to consume much more computation resources to guarantee the accuracy of each individual sample's soft label. [28] proposes Class-wise Self-Knowledge Distillation (CSKD), a more efficient approach to regularize the class-wise predictive distribution using a self-distillation fashion. Different from our method, [28] intends to encourage the consistency of soft labels within a training mini-batch, rather than learn optimal soft labels over the entire training set. We will demonstrate in Section 4.3 that, [1] does not outperform our method in the setting of self-distillation, where the student model has the same capacity with the teacher, and that, [28] is as efficient as our method, but less accurate.

4 Experiments

4.1 Tasks and Datasets

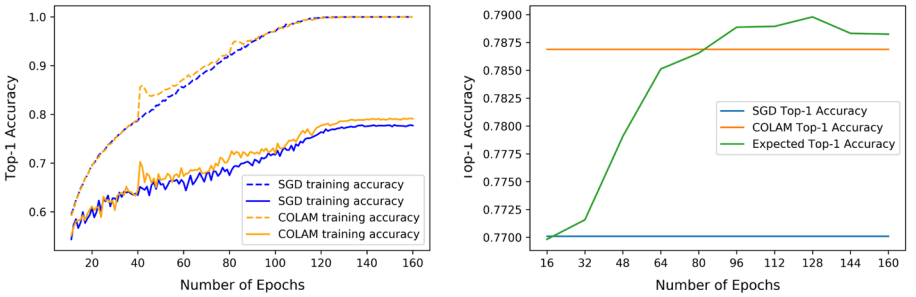
Image classification We use CIFAR10, CIFAR100 [13] and ImageNet [4] to test the performance of COLAM on image classification task. CIFAR10 has 10 different classes. In the training set of CIFAR10, each class contains 5,000 images. CIFAR 100 contains RGB images categorized into 100 classes, with each class composing 600 images. There are 500 training images and 100 testing images in each class. ImageNet is a tree-structured image database created according to the WordNet hierarchy. It consists of more than 20K categories and a total of 14 million images. We use the popular subset ILSVRC2012 which contains 1.3M images covering 1K categories.

Transfer learning We use ImageNet as the source dataset and 4 different target tasks covering typical types of plants, animals, objects and texture . The 4 target datasets are (1) Flower102 [18] which contains 102 categories of 8189 flower images, (2) Caltech-UCSD

Table 3 Test accuracy on ImageNet

Model	HL	LS [23]	CP [19]	DL [26]	LR [1]	CSKD [28]	COLAM
ResNet50	76.58±0.27	76.69±0.20	76.65±0.18	76.80±0.28	77.51±0.16	**77.21±0.16**	**77.56±0.25**
ResNet101	77.86±0.31	78.41±0.13	78.12±0.20	78.33±0.32	**78.75±0.21**	**78.40±0.27**	**78.75±0.18**

Same abbreviations are used as found in Table 1. LR refers to Label Refinery [1]. CSKD refers to Class-wise Self-Knowledge Distillation [28]. Note that LR costs *twice* the amount of training time as other methods



(a) learning curve **(b)** expected accuracy of the evolutionary soft labels

Fig. 1 Demonstration of the effect of COLAM through learning curve and expected accuracy. Experiments are performed on PreActResNet34. SGD accuracy refers to the test accuracy of models that are trained with hard labels using a vanilla SGD optimizer

Birds-200-2011 [25], which has 11,788 images classified into 200 categories, (3) FGVC-Aircraft [16] which composes 10,000 images of aircraft across 100 aircraft models, and (4) Describable Textures Dataset (DTD) [3] which is a texture database, consisting of 5640 images, organized according to a list of 47 terms (categories).

4.2 Settings

Image classification We train PreActResNet34 [7], WideResNet40x4 [29] and ResNeXt29_8x64d [27] for 160 epochs. We train DenseNet-BC-100 [10] for 240 epochs because it converges slower. The initial learning rate is 0.1 for all architectures. Training batch size is 64. We use standard SGD optimizer with momentum 0.9 and weight decay 0.0001. We apply standard data augmentation the same way as the Pytorch official examples on CIFAR10, CIFAR100 and ImageNet classification task. For CIFAR dataset, we pad the input images by 4 pixels, and then randomly crop a sub-region of 32×32 and randomly do a horizontal flip. For ImageNet, we first randomly crop a sub-region of 224×224 and randomly do a horizontal flip. We normalize the input data as done in common practice. We define *vanilla training* as the standard training procedure according to above settings without any label promoting strategy.For CIFAR10 and CIFAR100, we select the hyperparameters using 5-fold cross-validation on PreActResNet34 for each dataset, and set them fixed in all experiments. For ImageNet, we adopt the same method to decide the hyperparameters on ResNet50.

Transfer learning We use ResNet101 [6] as the base model to apply COLAM. We train the model with 40 epochs and the batch size for training is 64. SGD optimizer is used with a momentum of 0.9. The initial learning rate is set to 0.01 and the weight decay is set to 0.0001. We use exactly the same data augmentation methods as in ImageNet classification task. We repeat all these experiments 3 times and report the average Top-1 accuracy.

4.3 Results

Image classification Table 1 and 2 shows that our COLAM consistently and significantly improve baseline models in accuracy for the majority of neural network architectures we tested. On CFIAR100, the improvement is generally within 1%-2% in comparison to models trained with hard labels using a vanilla SGD optimizer. In tasks of CIFAR10 and CIFAR100,

Table 4 Test accuracy using ResNet101 on various datasets

Dataset	HL	LS [23]	COLAM
Flower102	0.9179	0.9279	**0.9294**
FGVC_Aircraft	0.7741	0.7675	**0.7787**
DTD	0.6646	0.6705	**0.6824**
CUB_200_2011	0.8172	0.8152	**0.8246**

All abbreviations follow the same rule as in Table 1

we find that models with more complex architectures are not guaranteed to be better than simpler ones. For example, WideResNet40x4 with 9 million parameters outperforms PreActResNet34 with 20 million parameters. This happens regardless of the training technique used. We can observe a similar improvement in Table 3 when we apply COLAM to different models on ImageNet. We also compare state-of-the-art knowledge distillation method [1] and [28] on ImageNet. Results in Table 3 show that our method is significantly superior to [28] on both ResNet-50 and ResNet-101. Compared against [1], which requires an additional training generation for teacher model training first, our method achieves comparable performance. The observation is consistent with that of authors in [1] that, benefits from knowledge distillation tend to be less significant when the student model is more powerful.

These results indicate the effectiveness of COLAM, which not only outperforms the traditional label smoothing technique, but it also beats other more dynamic but inherently equivalent form of label smoothing, namely Confidence Penalty and DisturbLabel. One reason that explains this phenomenon is that neither CP nor DisturbLabel encourages the model to learn the structural properties in the dataset when it regularizes the model. Preserving structural properties in the dataset is an important factor that contributes to good generalization of a model, as discussed in Sect. 5.2.

Transfer learning We fix our test model to be ResNet101 and perform experiments on the chosen datasets. Results in Table 4 indicate that, compared with models trained using hard labels, COLAM improves the transfer learning outcomes on all four datasets, and the improvements range from 0.46 to 1.78%. These results testify that COLAM can enhance model performance on varying datasets.

In contrast, LS does not always yield positively improved results. Furthermore, the extent of improvement LS brings is considerably less than that of COLAM, as shown by the statistics.

5 Discussion

5.1 Effect on Training Procedure

To further demonstrate the effectiveness of our proposed alternating optimization of the training loss and soft labels, we dive into the training process of COLAM. Experiments show how the evolution of soft labels helps learning.

We first plot the learning curve of the whole training procedure of PreActResNet34, as shown in Fig. 1a. For better demonstration purpose, we divide the training process into only 4 stages for COLAM training, with each stage consisting of 40 epochs. We observe that models trained with SGD using hard labels and COLAM display almost the same standard of performance in the first stage as expected. While in the 41st epoch, both training and test accuracy of COLAM get a sharp rise due to starting to involve soft labels generated in

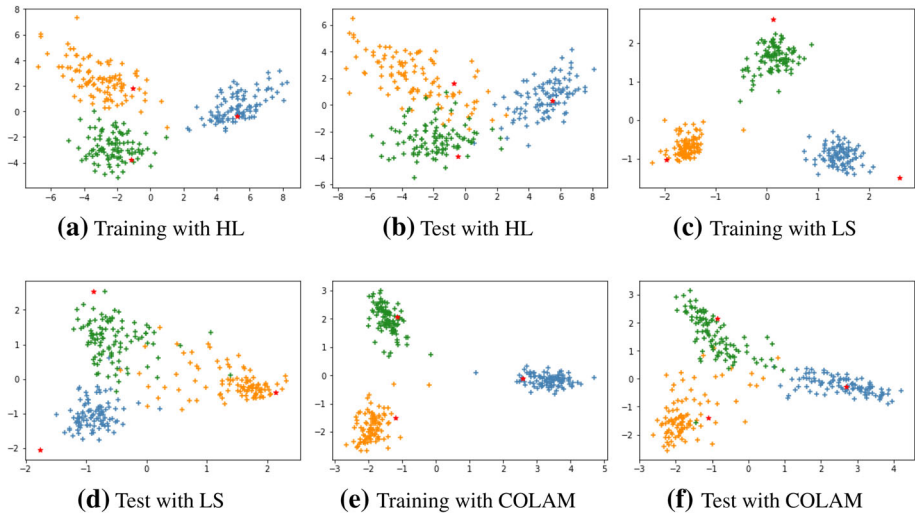


Fig. 2 Visualization of the penultimate layer representations and their relationships with the templates. The blue, orange and green clusters correspond to "man", "palm tree", and "pine tree" respectively. The red stars nearby are the templates of each cluster. "HL" abbreviates "hard labels."

the 40th epoch. Then both training and test accuracy values drop slightly for a few epochs, and then return to the trend of slowly rising for the remaining epochs until next stage. A similar phenomenon also appears at the beginning of next stage, although the magnitude of accuracy improvement becomes much smaller. We notice that since the first sharp rising, COLAM continuously outperforms training using hard labels on test set by a stable gap for the remaining epochs until convergence.

We do additional experiments to validate the effect of gradual promotion of soft labels, through our proposed alternative minimization approach. We divide the training process into 10 stages, each of which consists of 16 epochs. By performing COLAM, we obtain 10 checkpoints of soft labels. We train a model from scratch with vanilla SGD, except one difference that we use the supervision of these checkpoints. The top-1 accuracy of using each soft label is denoted as the corresponding "expected accuracy". It is a solid measure of the quality of a soft label. As illustrated in Fig. 1b, we observe that the expected accuracy gradually increases with the evolution of soft labels. In detail, the expected accuracy grows fast during the early period of the whole training procedure. This observation verifies that the quality of soft labels is indeed improved by alternating optimized with the training loss. It is worth noting that the expected accuracy begins to surpass COLAM after about half of the training epochs. The expected accuracy shows a slight drop near the end of the training process, indicating that alternating optimization of the two objectives may suffer from overfitting to a relatively small extent. Although COLAM improves the accuracy significantly, our experiments of the expected accuracy implies the potential existence of better design of soft labels.

5.2 Effect on Deep Representations

The empirical characteristics of COLAM observed in our experiments also show that COLAM is able to promote training by involving internal structural knowledge. We demonstrate this advantage through both qualitative and quantitative analysis.

Qualitative visualization Recent work [17] gives insights into why label smoothing improves model performance. They argue that the logit $\boldsymbol{x}^T \boldsymbol{w}_k$ for class k is correlated to the distance between $\boldsymbol{x}$ and $\boldsymbol{w}_k$, where $\boldsymbol{x}$ is the penultimate layer representation and $\boldsymbol{w}_k$ is the template for class k. As a result of their analysis, the penultimate layer representation $\boldsymbol{x}$ should be close to the correct class template $\boldsymbol{w}_k$ and equally distant from incorrect class templates $\boldsymbol{w}_i$ for all $i \neq k$ after a model is trained with LS.

Since our COLAM preserves the dataset structural properties compared to LS, $\boldsymbol{x}'$ is expected to be closer to the class template that shares a greater extent of inter-class similarity than a class that does not. We verify this by projecting both the penultimate layer representation and template in 2D. We choose 100 samples from each of three classes in CIFAR100 for this visualization: "man", "palm tree", and "pine tree." Intuitively, "palm tree", and "pine tree" should be more similar to each other than "man."

Fig. 2a and b show the cluster distributions on the training set and test set when ResNet56 is trained with hard labels using vanilla SGD. We observe that the clusters are close to their respective templates. However, they are generally spread out and scattered. The clusters of "palm tree" and "pine tree" are relatively closer compared with "man." This reflects the structural property within the dataset.

As seen in Fig. 2c and d, the clusters of label smoothing are tighter and easier to separate. The three clusters also try to be equally distant from the other classes' templates, resulting in a situation where clusters are drawn inward closer to the center of the subspace formed by the templates. However, the structural properties of the dataset is no longer preserved. "palm tree" and "pine tree" have the same distance as that between "palm tree" and "man."

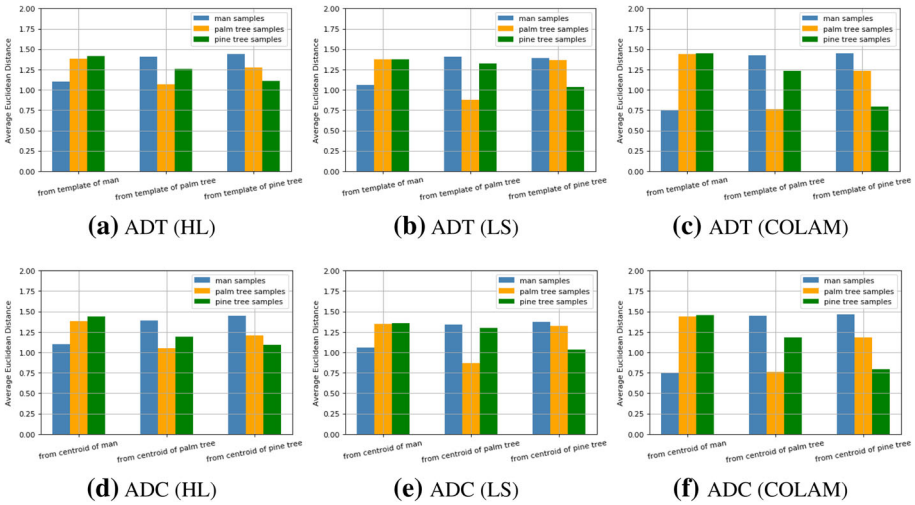
Next, we see in Fig. 2e and f that, when ResNet56 is trained with COLAM, the clusters are better separated in both training set and test set. Each cluster remains close to its own template, but further away from the center of the subspace this time. This indicates the model's improved ability to distinguish each sample. Additionally, each cluster looks tighter in comparison to the clusters in Fig. 2a and b. What remains unchanged is the structural properties in the dataset: "palm tree" and "pine tree" are still closer and "man" is further away from these two. In comparison, LS does not maintain this structural property.

Observing all figures as a whole, we see that our method COLAM is a "neutralizer" between using hard labels and LS. COLAM enjoys both accurately representing dataset structural properties (shown in training with hard labels) and obtaining tighter clusters that are easier for classification (shown in training with LS). This "neutralizing" effect enables a model to better generalize.

Quantitative evaluation We quantitatively evaluate three distance criteria for the same three classes to confirm the qualitative findings described above. Specifically, we find

- The Euclidean distance between the normalized templates, shown in Table 5.
- The average Euclidean distance between the template and samples' penultimate layer representations, shown in Fig. 3a–c.
- The average Euclidean distance between the centroid and samples' penultimate layer representations, shown in Fig. 3d–f.

Note that we normalize all vectors when we compute such distances in order to make fair comparisons.



(a) ADT (HL) **(b)** ADT (LS) **(c)** ADT (COLAM)

(d) ADC (HL) **(e)** ADC (LS) **(f)** ADC (COLAM)

Fig. 3 Clustering effects: Average distance from each template (ADT)/centroid (ADC) of the cluster to samples. Fig. 3a–c show Average distance from each template to samples while Fig. 3d–f show that from each centroid. "HL" abbreviates "hard labels". "LS" abbreviates "label smoothing"

Table 5 Euclidean distance between the normalized templates

Distance	HL	LS	COLAM
(Man, Palm Tree)	1.433	1.460	1.438
(Man, Pine Tree)	1.425	1.400	1.461
(Palm Tree, Pine Tree)	1.264	1.369	1.227
Average	1.374	1.410	1.375

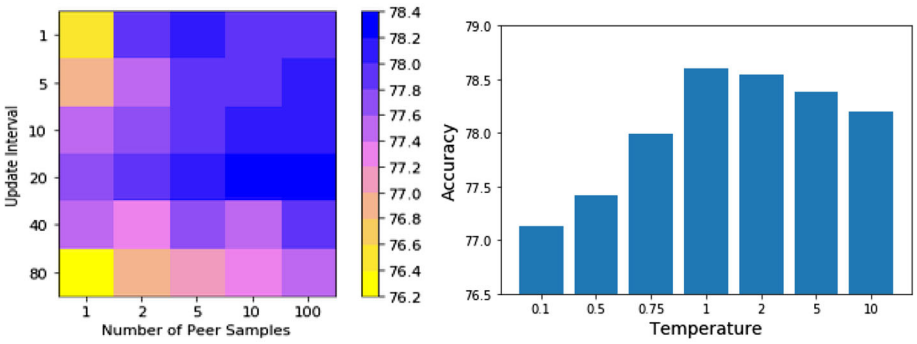
Same abbreviations are used as found in Table 1

As shown in Table 5, COLAM largely preserves the structural properties in the dataset by keeping the distance between "palm tree", and "pine tree" templates closer and distance from "man" greater, which is consistent with the model trained with hard labels.

In contrast, LS generally enlarges the overall distance in between templates, due to the requirement that each cluster should be equi-distant away from the incorrect class templates. Since clusters will stick close to their respective templates after training, this observation implies that data structural properties is missing in LS.

Figure 3a–c shows the distance between templates and clusters. We see that COLAM (Fig. 3c) is able to preserve structural properties in the dataset because the distance from a template to other clusters display the same trend as that in Fig. 3a. Together, the two figures also show that each template is closer to its corresponding cluster and further away from other clusters when the model is trained with COLAM. Since Table 5 shows that the distance between the templates in these two methods are roughly the same, the difference between *the average distance from a template to its own cluster* and *the average distance from a template to other clusters* indicates how well the clusters are separated. Figure 3c shows a larger such difference than Fig. 3a does. This explains why COLAM outperforms training with hard labels.

Now we compare COLAM (Fig. 3c) with LS (Fig. 3b). Because COLAM does not need to force each cluster to be equi-distant away from the incorrect class templates and preserves



(a) # peer samples vs. # epochs per stage **(b)** Sensitivity of T

Fig. 4 COLAM hyperparameter experiments on PreActResNet18

structural properties of the data, the distance between a template to its own cluster can get far smaller than that in LS. This smaller distance contributes to a larger difference between *the average distance from a template to its own cluster* and *the average distance from a template to other clusters*. Hence, COLAM is able to improve model generalization to a greater extent than LS. Moreover, clusters in LS are almost equi-distant away from other classes' templates as shown in Fig. 3b, this violates the structural properties in the dataset.

Computing the average distance between samples and their centroid reveals how tight a cluster is. A smaller average value indicates a tighter cluster and vice versa. In Fig. 3d–f, we observe the smallest such distance is found in the model trained with COLAM.

We are also interested in the difference between *the distance from a centroid to its own cluster* and *the distance from a centroid to other clusters*. The larger this difference is, the better the clusters are separated. Figure 3f indicates that the model trained with COLAM yields the greatest such distance. This, from a slightly different view, explains why COLAM gives rise to the highest generalization ability.

5.3 Choice of Hyperparameters

The most important two hyperparameters are the number of stages and number of random peer samples. We use update intervals, or equivalently number of epochs per stage, instead of number of stages for clarity in this experiments. We set update intervals to be [1, 5, 10, 20, 40, 80] and number of peer samples to be [1, 2, 5, 10, 100]. We run a grid search method to validate different combinations of these two variables. Note that the evaluation directly performed on the test set aims at illustrating the sensitivity of hyperparameters, rather than hyperparameter selection.

In Fig. 4a we see that performance of COLAM does not seem to be very sensitive to most combinations of the hyperparameters. When the number of peer samples increases to 5 or more, model accuracy tends to be over 77.3%. Even when the number of peer samples is low, a good choice of the value of update interval can boost the model performance significantly. Low accuracy of the model only happens consistently when the value of update interval is large.

Another important hyperparameter is the temperature T, which softens the probability distribution of incorrect classes of soft labels. We explore $T = [0.1, 0.5, 0.75, 1, 2, 5, 10]$

used in PreActResNet34 on CIFAR100. Theoretically, a larger value of T makes the probability distribution of the incorrect classes smoother. When T gets sufficiently large, this will make our COLAM to behave like traditional LS. In contrast, the probability distribution among incorrect classes gets even sharper if T is less than 1. When T gets close to 0, it is closed to the original hard label. Empirically we recommend T to be some value between 1 to 2.

6 Conclusion

In this paper, we have discussed the advantages of using soft labels as the target in deep learning and proposed a novel algorithm COLAM that alternatively minimizes the training loss subject to the soft label and the objective to learn improved soft labels. We have conducted numerous experiments to demonstrate the method's effectiveness on a variety of tasks. We have also offered both qualitative and quantitative explanations as to why COLAM is more beneficial than existing techniques to produce soft labels.

References

1. Bagherinezhad H, Horton M, Rastegari M, Farhadi A (2018) Label refinery: improving imagenet classification through label progression. arXiv:1805.02641
2. Chorowski J, Jaitly N (2016) Towards better decoding and language model integration in sequence to sequence models. In: INTERSPEECH
3. Cimpoi M, Maji S, Vedaldi A (2015) Deep filter banks for texture recognition and segmentation. CVPR, pp 3828–3836
4. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. CVPR, pp 248–255
5. Graves A, Jaitly N (2014) Towards end-to-end speech recognition with recurrent neural networks. In: ICML
6. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. CVPR, pp 770–778
7. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. arXiv:1603.05027
8. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. arXiv:1503.02531
9. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580
10. Huang G, Liu Z, Weinberger KQ (2016) Densely connected convolutional networks. CVPR, pp 2261–2269
11. Huang Y, Cheng Y, Chen D, Lee H, Ngiam J, Le QV, Chen Z (2018) Gpipe: Efficient training of giant neural networks using pipeline parallelism. arXiv:1811.06965
12. Józefowicz R, Vinyals O, Schuster M, Shazeer N, Wu Y (2016) Exploring the limits of language modeling. arXiv:1602.02410
13. Krizhevsky A (2009) Learning multiple layers of features from tiny images
14. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: NIPS
15. Lopez-Paz D, Bottou L, Schölkopf B, Vapnik V (2016) Unifying distillation and privileged information. Int Conf Learn Represent (ICLR)
16. Maji S, Rahtu E, Kannala J, Blaschko MB, Vedaldi A (2013) Fine-grained visual classification of aircraft. arXiv:1306.5151
17. Müller R, Kornblith S, Hinton GE (2019) When does label smoothing help? CoRR arXiv:1906.02629
18. Nilsback ME, Zisserman A (2008) Automated flower classification over a large number of classes. ICVGIP, pp 722–729
19. Pereyra G, Tucker G, Chorowski J, Kaiser L, Hinton GE (2017) Regularizing neural networks by penalizing confident output distributions. arXiv:1701.06548
20. Real E, Aggarwal A, Huang Y, Le QV (2018) Regularized evolution for image classifier architecture search. In: AAAI

21. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. CoRR arXiv:1409.1556
22. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2014) Going deeper with convolutions. CVPR
23. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: CVPR, pp 2818–2826
24. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: NIPS
25. Wah C, Branson S, Welinder P, Perona P, Belongie S (2011) The caltech-UCSD birds-200-2011 dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology
26. Xie L, Wang J, Wei Z, Wang M, Tian Q (2016) Disturblabel: regularizing cnn on the loss layer. CVPR, pp 4753–4762
27. Xie S, Girshick RB, Dollár P, Tu Z, He K (2016) Aggregated residual transformations for deep neural networks. CVPR, pp 5987–5995
28. Yun S, Park J, Lee K, Shin J (2020) Regularizing class-wise predictions via self-knowledge distillation. In: The IEEE/CVF conference on computer vision and pattern recognition (CVPR)
29. Zagoruyko S, Komodakis N (2016) Wide residual networks. arXiv:1605.07146
30. Zeiler MD, Fergus R (2013) Stochastic pooling for regularization of deep convolutional neural networks. CoRR arXiv:1301.3557
31. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O (2016) Understanding deep learning requires rethinking generalization. arXiv:1611.03530
32. Zhang G, Wang C, Xu B, Grosse RB (2018) Three mechanisms of weight decay regularization. arXiv:1810.12281
33. Zoph B, Vasudevan V, Shlens J, Le QV (2017) Learning transferable architectures for scalable image recognition. CVPR, pp 8697–8710