Knowledge Refinery: Learning from Decoupled Label

Qianggang Ding[1], **Sifan Wu**[1], **Tao Dai**[1,2,*], **Hao Sun**[3], **Jiadong Guo**[4,5],
Zhang-Hua Fu[6,7], **Shutao Xia**[1,2]
[1] Tsinghua University, [2] PCL Research Center of Networks and Communications, Peng Cheng Laboratory,
[3] The Chinese University of Hong Kong, [4] The Hong Kong University of Science and Technology,
[5] International Digital Economy Academy, [6] The Chinese University of Hong Kong, Shenzhen,
[7] Shenzhen Institute of Artificial Intelligence and Robotics for Society
{dqg18, wusf18}@tsinghua.edu.cn, daitao.edu@gmail.com, sh018@ie.cuhk.edu.hk, gjd@idea.edu.cn,
fuzhanghua@cuhk.edu.cn, xiast@sz.tsinghua.edu.cn

Abstract

Recently, a variety of regularization techniques have been widely applied in deep neural networks, which mainly focus on the regularization of weight parameters to encourage generalization effectively. Label regularization techniques are also proposed with the motivation of softening the labels while neglecting the relation of classes. Among them, the technique of knowledge distillation proposes to distill the soft label, which contains the knowledge of class relations. However, this technique needs to pre-train an extra cumbersome teacher model. In this paper, we propose a method called Knowledge Refinery (KR), which enables the neural network to learn the relation of classes on-the-fly without the teacher-student training strategy. We propose the definition of decoupled labels, which consist of the original hard label and the residual label. To exhibit the generalization of KR, we evaluate our method in both fields of computer vision and natural language processing. Our empirical results show consistent performance gains under all experimental settings.

Introduction

Deep neural networks (DNNs) have recently been widely and successfully used in various tasks due to their powerful capabilities of extracting features. However, DNNs often contain millions of trainable parameters which easily cause the over-fitting problem. To solve this problem, many regularization methods including regularization on parameters (e.g., Dropout (Srivastava et al. 2014)), regularization on data (e.g., data augmentation), and regularization on labels (e.g., label smoothing(Srivastava, Greff, and Schmidhuber 2015)) have been developed to avoid over-fitting problems. Among them, parameter regularization methods need to modify and retrain DNN models, while data regularization methods require collecting a huge amount of training samples, both of which limit the practical value of DNNs in reality.

By contrast, label regularization methods (Szegedy et al. 2016; Pereyra et al. 2017; Bagherinezhad et al. 2018) are performed in a more friendly way and have recently received much attention, as they do not need to retrain/modify network

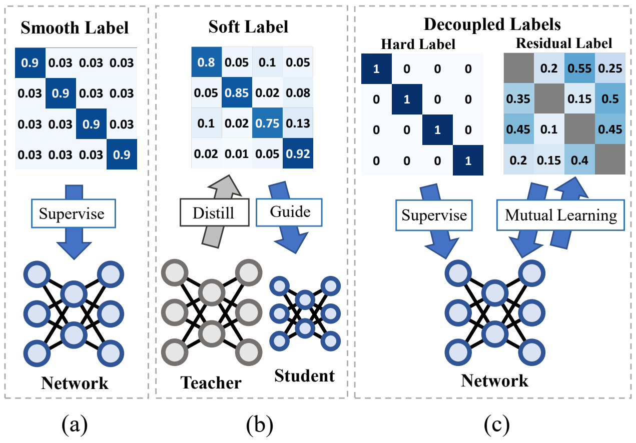


Figure 1: (a) Label by label smoothing, (b) Label by knowledge distillation, (c) Label by knowledge refinery. Label smoothing neglects the class-level relation. Knowledge distillation need to pre-train a cumbersome teacher model. Instead, our method incorporates the relation of classes by decoupled labels WITHOUT pre-training a teacher model.

structures. The general idea is that one-hot encoded labels are so hard which make DNN suffer from the over-fitting problem. In general, these methods can be further divided into two types of methods, *label smoothing* (Szegedy et al. 2016) and *label softening* based methods (Hinton, Vinyals, and Dean 2015). Label smoothing reduces the confidence score by uniformly redistributing 10% of the weight from the ground-truth class to incorrect classes, while neglecting the knowledge of class-level relations (see Fig. 1(a)). For instance, an image of a "dog" that has similar visual appearance to a "cat" should have a higher label score than an image of a "plane" that has different visual appearance. To make labels more informative, label softening based methods attempt to extract the visual relations between classes to impose such knowledge into the training phase. Typical methods of this type are knowledge distillation methods (Hinton, Vinyals, and Dean 2015) that generate soft labels from a cumbersome teacher model to train a small student model (see Fig. 1(b)).

[*]Corresponding author: Tao Dai.

Although these existing label regularization methods relieve the over-confidence problem with soft labels, they may suffer from the under-confidence problem when the target label is over-smoothed. Therefore, they all have to adjust trade-off hyper-parameters to avoid over-/under-confidence problems. Overall, we can conclude that the "ideal" label regularization method should satisfy the following conditions: (i) *Relation expression*. The representation of labels consider the relation of classes. (ii) *Online training*. Learning the relation of classes on-the-fly without pre-training an extra teacher model. (iii) *Flexibility*. There is no need to adjust the softness of label representation.

To this end, we propose a novel label regularization method, called Knowledge Refinery (KR), by learning the knowledge of class-specific information and class-level relations dynamically. We decouple the "ideal" label into two components, the target label (i.e. the hard label or the smooth label) and the non-target label (i.e. the residual label), as is shown in Fig. 1(c). More specifically, instead of directly updating the target label, we update the residual label during training iteratively by the proposed knowledge refinery regularizer module. Compared with the technique of knowledge distillation (KD) which requires an extra teacher model, KR produces dynamical and informative labels on-the-fly. Overall, KR satisfies all the above conditions of the "ideal" label regularization method — *Relation expression*, *Online training*, and *Flexibility*. Furthermore, KR can serve as a universal regularizer and can be equipped into arbitrary vanilla deep neural networks.

The main contributions can be summarized as follows:

- We propose a new label regularization method, called knowledge refinery (KR), to enable networks to learn more informative knowledge from decoupled labels, which consists of *hard label* and *residual label*.
- We propose a mutual learning strategy to learn the relation of classes on-the-fly without teacher models. The extra memory and time consumption of our method are negligible in practice.
- The proposed KR regularizer serves as a plug-n-play module and can be equipped into an arbitrary vanilla DNN. The comprehensive experiments on different public benchmark datasets for image/text classification tasks verify the effectiveness of our method.

Related Works

Knowledge Distillation

Previous research on knowledge distillation (Hinton, Vinyals, and Dean 2015; Lopez-Paz et al. 2015; Zhu, Gong et al. 2018; Zhang et al. 2018; Wang, Liu, and Tao 2020) explored the transmission from deep and cumbersome teacher models to shallow and light student models. The rationale behind this technology is to ensemble the information distilled by one or more teacher models as extra supervision for the student model. Hinton et al. (2015) first applied soft labels produced by a teacher model to train a small student model. The soft label refined by the teacher model is defined as $q_i^{(soft)} = \frac{exp(z_i/T)}{\sum_{j=1}^{K} exp(z_j/T)}$, where K is the number of categories and T is a temperature that controls the smoothness of the soft label. With the technique of knowledge distillation, the student model can learn from the cumbersome teacher model and therefore improves the performance. In contrast, our method is more efficient without the teacher-student training strategy, and the hyper-parameter in our loss function is insensitive in practice.

Label Regularization

There exist different types of regularization methods to improve the generalization of neural networks, including (i) regularization on model parameters like Dropout (Srivastava et al. 2014; Zoph et al. 2018; Wan et al. 2013), (ii) regularization on model inputs like data augmentation, (iii) regularization on features like batch normalization and group normalization (Ioffe and Szegedy 2015), and (iv) regularization on labels like label regularization (Szegedy et al. 2016; Xie et al. 2016; Lee et al. 2018; Müller, Kornblith, and Hinton 2019; Xu et al. 2020; Chen et al. 2020). Among them, regularization methods on model parameters need to change the network structure and retrain the model, while they model inputs need to create a huge number of training samples. In contrast, regularization methods on labels are more efficient and train-friendly without the need of modifying network structures/inputs. In contrast to these methods, our method regularizes labels by learning the relations of labels using negligible extra computation consumption.

Preliminary

In this paper, we consider a K-class classifier $h_\theta(\boldsymbol{x}) : \mathcal{X} \to \mathcal{C}$, parameterized by θ, where $\mathcal{X}$ is the input space and $\mathcal{C} = \{1, ..., K\}$ is the label space. Assuming the classifier $h = g \circ f$ can be decomposed of two embedding functions, one is $f_\theta : \mathcal{X} \to \mathcal{Z}$, which convert input features to logit vectors $\boldsymbol{z} = f_\theta(\boldsymbol{x})$, and the other one is the softmax layer $g : \mathcal{Z} \to \mathcal{C}$, which convert logit vectors to the specific category. The cross-entropy loss (CE) function is $\mathcal{L}_{CE}(\boldsymbol{p}, \boldsymbol{q}) = -\sum_{i=1}^{K} p_i \log q_i$, where $\boldsymbol{p} = softmax(\boldsymbol{z})$, and $\boldsymbol{q}$ is the label vector like the one-hot encoded vector. Kullback-Leibler (KL) divergence is $D_{KL}(\boldsymbol{p}||\boldsymbol{q}) = \sum_{i=1}^{K} p_i \log(\frac{p_i}{q_i})$. Jensen-Shannon (JS) divergence is $D_{JS}(\boldsymbol{p}||\boldsymbol{q}) = \frac{1}{2} D_{KL}(\boldsymbol{p}||\frac{\boldsymbol{p}+\boldsymbol{q}}{2}) + \frac{1}{2} D_{KL}(\boldsymbol{q}||\frac{\boldsymbol{p}+\boldsymbol{q}}{2})$.

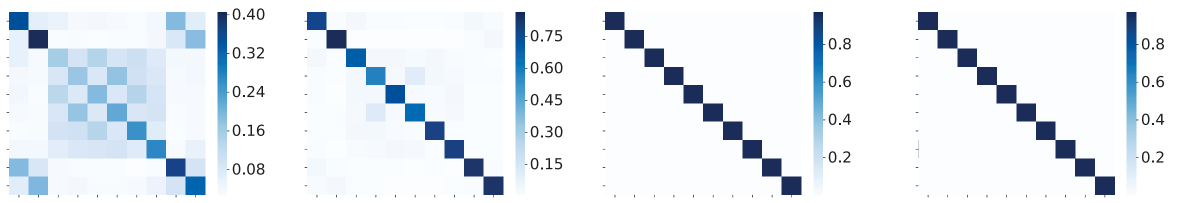
Rethink Knowledge Distillation

In this section, we introduce some observations which motivate us to decouple the label. Let us review the loss function by knowledge distillation method (Hinton, Vinyals, and Dean 2015), which add a KL divergence loss term to the original cross-entropy loss function as follows:

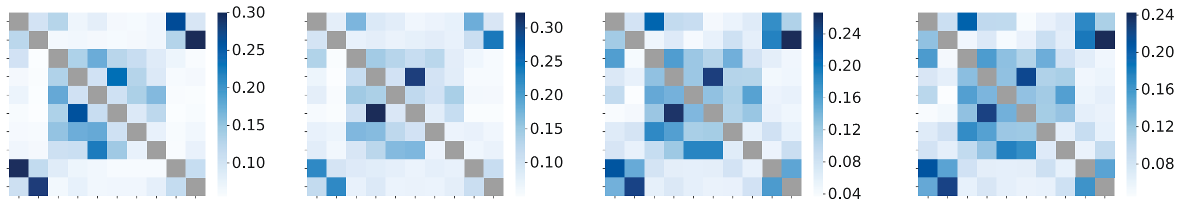
$$\mathcal{L}_{KD} = (1-\alpha)\mathcal{L}_{hard} + \alpha T^2 \mathcal{L}_{soft} \tag{1}$$

$$\text{where } \mathcal{L}_{hard} = \mathcal{L}_{CE}(\boldsymbol{p}, \boldsymbol{q}) \text{ and } \mathcal{L}_{soft} = D_{KL}(\boldsymbol{p}||\boldsymbol{q}^{(\boldsymbol{soft})}) \tag{2}$$

where α is a trade-off parameter and T is the temperature factor. Let us rethink this loss function $\mathcal{L}_{KD}$. Given a model to recognize images belong to three classes "cat", "dog",



(a) 1^{st} epoch ($\mu = 44.12\%$) (b) 10^{th} epoch ($\mu = 84.60\%$) (c) 200^{th} epoch ($\mu = 99.99\%$) (d) 300^{th} epoch ($\mu = 99.99\%$)



(e) 1^{st} epoch ($\mu = 44.12\%$) (f) 10^{th} epoch ($\mu = 84.60\%$) (g) 200^{th} epoch ($\mu = 99.99\%$) (h) 300^{th} epoch ($\mu = 99.99\%$)

Figure 2: Visualization of the learned soft labels and residual labels: (a)(b)(c)(d) represent the soft label, (e)(f)(g)(h) represent the corresponding residual label with regard to (a)(b)(c)(d), respectively. μ denotes the training accuracy after the specific epoch. The captions of the x-axis from top to bottom and of the y-axis from left to right are: plane, car, bird, cat, deer, dog, frog, horse, ship, ship, and truck.

"plane". Supposing the soft label $\boldsymbol{q}^{(soft)}$ of this "cat" image is $[0.6, 0.3, 0.1]$, which is learned by the teacher model. And suddenly the confidence probabilities $\boldsymbol{p}$ predicted by the model for this image is $[0.7, 0.2, 0.1]$. As a result, we know that $\mathcal{L}_{soft}$ would force the output $\boldsymbol{p}$ close to the soft label $\boldsymbol{q}$. That is, $\mathcal{L}_{soft}$ attempt to decrease the confidence probability of "cat" from 0.7 to 0.6. By contrast, $\mathcal{L}_{hard}$ (i.e. the cross-entropy loss based on the one-hot encoded label) would force the confidence probability of "cat" from 0.7 close to 1.0. We can naturally find that the optimization objective of $\mathcal{L}_{soft}$ and $\mathcal{L}_{hard}$ in the loss function $\mathcal{L}_{KD}$ is contradictory in this case. In other words, the model trained with $\mathcal{L}_{KD}$ may struggle with the balance of under- and over-confidence problem.

We also observed the learned soft labels and got some interesting findings. Take CIFAR-10 dataset for an example, we obtain the soft labels of all categories by the following steps: (i) We save the predicted probability $\boldsymbol{p}_i = softmax(\boldsymbol{z})$ for each sample i, (ii) Calculate the soft label $\hat{\boldsymbol{q}}_k$ of category k by $\hat{\boldsymbol{q}}_k = \frac{1}{|\mathcal{T}_k|}\sum_{i\in\mathcal{T}_k}\boldsymbol{p}_i$, where $\mathcal{T}_k$ denotes the set of samples of category k. Afterwards, we can obtain the soft labels of K categories and each $\hat{\boldsymbol{q}}_k$ is a vector of size K. That means we can use a matrix of size $K \times K$ to represent these soft labels. In Fig. 2 (the first row), we give the visualization of the soft labels of categories learned after 1, 10, 200, and 300 epochs. As we can see, the soft labels become harder with regard to the increasing epochs, and finally the probabilities of the ground-truth labels (diagonal) almost dominate the whole heatmap. To expose the probabilities of non-ground-truth labels (non-diagonal), we attempt to mask the ground-truth position k in the calculation of softmax function as follows:

$$\boldsymbol{z}^{(res)} = (\boldsymbol{z})_{\overline{k}}, \tag{3}$$

$$\boldsymbol{p}^{(res)} = softmax(\boldsymbol{z}^{(res)}), \tag{4}$$

where $(\boldsymbol{z})_{\overline{k}}$ means to mask the k-th position of $\boldsymbol{z}$. The masked probability $\boldsymbol{p}^{(res)}$ is called *residual output* in this paper. Afterwards, we use the same strategy to calculate the masked soft labels (i.e. *residual label*) of categories on $\boldsymbol{p}^{(res)}$ as $\hat{\boldsymbol{q}}_k^{(res)} = \frac{1}{|\mathcal{T}_k|}\sum_{i\in\mathcal{T}_k}\boldsymbol{p}_i^{(res)}$, and the visualization can be seen in Fig. 2 (the second row). As a result, we can surprisingly find that (i) the heatmaps keep soft consistently from the beginning to end, (ii) the heatmaps have fairly well consistency, which means that the relation of labels can be learned fairly well even after only one epoch. Based on these findings, we think that the masked soft label is a better representation than the soft label.

Overall, the non-diagonal disappearance of the heatmaps in the first row and the softness consistency of the heatmaps in the second row motivated us to explore the masked soft label (i.e. *residual label*). To this end, we decouple the original label to two components — *hard label* and *residual label*. Here the hard label means the original one-hot encoded label which represents the specific-class information, and the residual label means the masked soft label which represents the class-level relations. The main advantages of decoupled labels include the following aspects: (i) there is no need to set hyper-parameter to trade-off the hard loss and the soft loss since the hard label and the residual label can complement each other well. (ii) the softness consistency allows us to design a regularization strategy for the residual label. We will introduce the details of our proposed regularizer in later sections.

Knowledge Refinery

Our *Knowledge Refinery* (KR) method aims to regularize the neural networks on-the-fly during the training phase. In

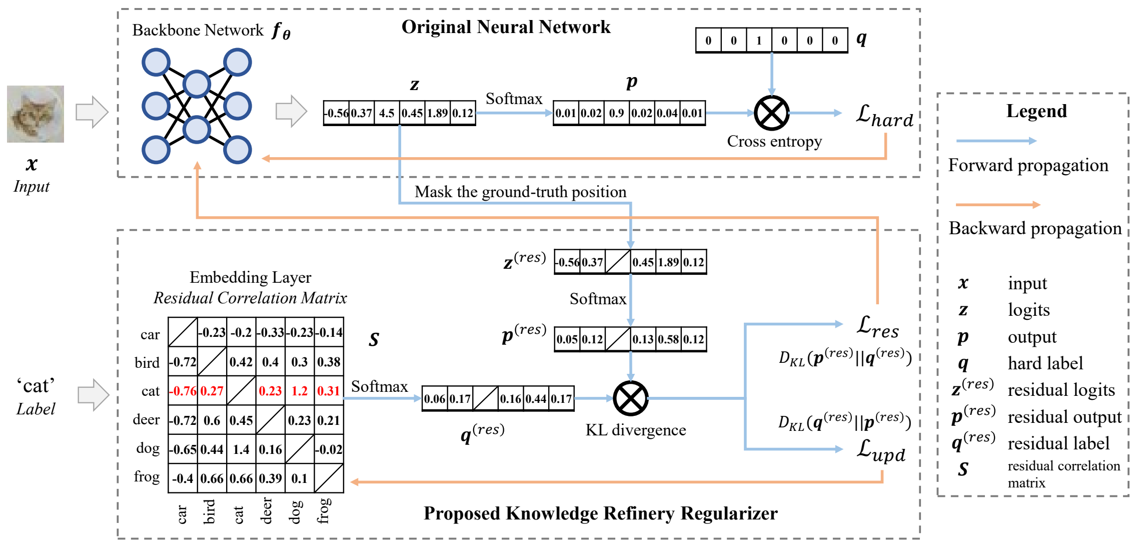


Figure 3: Pipeline of Knowledge Refinery with decoupled labels — the residual label $\boldsymbol{q}^{(res)}$ and the hard label $\boldsymbol{q}$.

general, KR can be treated as a plug-n-play regularizer module, which can be unified into an arbitrary vanilla DNN, such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG(Simonyan and Zisserman 2014), and ResNet (He et al. 2016a), as is shown in Fig. 3.

Workflow

The KR framework briefly consists of two modules — *Original Neural Network* and *Knowledge Refinery Regularizer*.

Original Neural Network The module of Original Neural Network is a vanilla neural network supervised by one-hot encoded labels (i.e. hard label). The loss of this module is the cross-entropy loss between the predicted probabilities $\boldsymbol{p}$ and one-hot encoded labels $\boldsymbol{q}$, which is typically as $\mathcal{L}_{hard} = \mathcal{L}_{CE}(\boldsymbol{p}, \boldsymbol{q})$ (i.e. hard loss).

Knowledge Refinery Regularizer The module of Knowledge Refinery Regularizer works on the logits $\boldsymbol{z}$ outputted by Original Neural Network. Firstly, recall the definition of *residual label* in previous section, we mask the k-th position of $\boldsymbol{z}$, where k represents the target class index. Next, as is shown in Eq. 3 and Eq. 4, we normalize the masked logits $\boldsymbol{z}^{(res)}$ using the softmax function, and then we obtain the predicted residual probability $\boldsymbol{p}^{(res)}$. On the other side, we adopt an embedding layer whose weight is a matrix $\boldsymbol{S} \in \mathbb{R}^{K\times K}$, which represents *Residual Correlation Matrix*. Then we can directly obtain the residual label of category k by adopting softmax normalization on the k-th row of $\boldsymbol{S}$ as follows:

$$\boldsymbol{q}^{(res)} = softmax(\boldsymbol{S}_k). \tag{5}$$

The loss of Knowledge Refinery Regularizer composes of *residual loss* and *update loss*. More specifically, the residual loss $\mathcal{L}_{res}$ and the update loss $\mathcal{L}_{upd}$ is a pair of mutual learning losses, the definition of which is as follows:

$$\mathcal{L}_{upd} = D_{KL}(\boldsymbol{q}^{(res)}||\boldsymbol{p}^{(res)}), \tag{6}$$

$$\mathcal{L}_{res} = D_{KL}(\boldsymbol{p}^{(res)}||\boldsymbol{q}^{(res)}), \tag{7}$$

$$\mathcal{L}_{mut} = \mathcal{L}_{upd} + \mathcal{L}_{res}. \tag{8}$$

Note that we initialize *Residual Correlation Matrix* to zero-filled matrix. That means the initial residual labels of categories are uniform distributions. The overall workflow can refer to Alg. 1. By this mutual learning strategy, we allow the backbone network and Residual Correlation Matrix learn from each other simultaneously. Moreover, recall the observed softness consistency of the residual label which is mentioned before, this mutual learning strategy also enable the model to avoid "consistency shift" by stabilizing the learned residual probabilities $\boldsymbol{p}^{(res)}$.

Backward Propagation

In the mutual learning strategy, we aim to use $\mathcal{L}_{upd}$ to update the residual labels derived from $\boldsymbol{S}$ and use $\mathcal{L}_{res}$ to regularize the backbone network f_θ via the learned residual probabilities $\boldsymbol{p}^{(res)}$. Therefore, we block the useless gradients in $\mathcal{L}_{upd}$ and $\mathcal{L}_{res}$ in the backward-propagation phase, respectively. Specifically, we propagate the gradient of $\mathcal{L}_{upd}$ only to the embedding layer $\boldsymbol{S}$ by blocking the gradient of $\boldsymbol{p}^{(res)}$ in Eq. 6. Same for the backward-propagation of $\mathcal{L}_{res}$, we block the gradient of $\boldsymbol{q}^{(res)}$ in Eq. 7 so that $\mathcal{L}_{res}$ only affects the backbone network f_θ. With this backward-propagation mechanism, $\boldsymbol{p}^{(res)}$ and $\boldsymbol{q}^{(res)}$ can be learned mutually. The overall loss is simply the linear summation of $\mathcal{L}_{hard}$ and $\mathcal{L}_{mut}$ as follows:

$$\mathcal{L}_{tot} = \mathcal{L}_{hard} + \alpha\mathcal{L}_{mut}. \tag{9}$$

Objective

In this section, we respectively analyze the objective of the backbone network f_θ and Residual Correlation Matrix $\boldsymbol{S}$ based on the aforementioned backward-propagation mechanism. For the backbone network f_θ, the objective can be

Algorithm 1 Workflow of Knowledge Refinery

```
1:  $\boldsymbol{S} \leftarrow \boldsymbol{0}$   ▷ Initialize the residual correlation matrix with zero matrix.
2:  for $epoch = 1, 2, \ldots$ do
3:      $\boldsymbol{z} = f_\theta(\boldsymbol{x})$   ▷ Here we omit mini-batch for simplicity.
4:      $\boldsymbol{p} = softmax(\boldsymbol{z})$
5:      $\mathcal{L}_{hard} = \mathcal{L}_{CE}(\boldsymbol{p}, \boldsymbol{q})$   ▷ Calculate the hard loss.
6:      $\boldsymbol{z}^{(res)} = (\boldsymbol{z})_{\overline{k}}$   ▷ $k$ is the index of the ground-truth class.
7:      $\boldsymbol{p}^{(res)} = softmax(\boldsymbol{z}^{(res)})$
8:      $\boldsymbol{q}^{(res)} = softmax(\boldsymbol{S}_k)$
9:      $\mathcal{L}_{upd} = D_{KL}(\boldsymbol{q}^{(res)}||\boldsymbol{p}^{(res)})$   ▷ Calculate the update loss.
10:     $\mathcal{L}_{res} = D_{KL}(\boldsymbol{p}^{(res)}||\boldsymbol{q}^{(res)})$   ▷ Calculate the residual loss.
11:     $\theta \leftarrow \theta - \beta\nabla\mathcal{L}_{hard} + \alpha\mathcal{L}_{res}$   ▷ Optimize the backbone network.
12:     $\boldsymbol{S} \leftarrow \boldsymbol{S} - \beta\nabla\alpha\mathcal{L}_{upd}$   ▷ Optimize the residual correlation matrix.
13: end for
```

formulated as follows:

$$\arg\min_{\theta}(\mathcal{L}_{hard} + \alpha\mathcal{L}_{res}). \tag{10}$$

On the other side, the objective of Residual Correlation Matrix $\boldsymbol{S}$ can be formulated as follows:

$$\arg\min_{S}(\alpha\mathcal{L}_{upd}), \tag{11}$$

where α is a hyper-parameter. In practice, we set the hyper-parameter α to a exponentially decreasing function $\alpha = \gamma^t$, where t denotes the number of epochs and γ is the decay factor which is set to 0.99 empirically. As is mentioned in previous section, the hard loss and the residual loss can be complement each other well, therefore there is no need to trade-off these two losses. Our experiments also show that the value of γ is insensitive to the performance in the later section. For the theoretical analysis on the effectiveness of knowledge refinery, please refer to Appendix A.

Label Smoothing Enhanced KR

Label smoothing (Szegedy et al. 2016) proposed a mechanism for encouraging the model to be less confident about ground-truth to improve generalization. Typically, the smooth label $\boldsymbol{q}^{(smooth)}$ gives 10% confidence of the ground-truth class to incorrect classes uniformly as

$$\begin{cases} q_i^{(smooth)} = 0.9 & i = k, \\ q_i^{(smooth)} = 0.1 & i \neq k, \end{cases} \tag{12}$$

In our method, we can easily apply the technique of label smoothing by replacing the hard label to the smooth label. After enhancing by label smoothing, the over loss Eq. 9 will be changed as

$$\mathcal{L}_{tot} = \mathcal{L}_{smooth} + \alpha\mathcal{L}_{mut}. \tag{13}$$

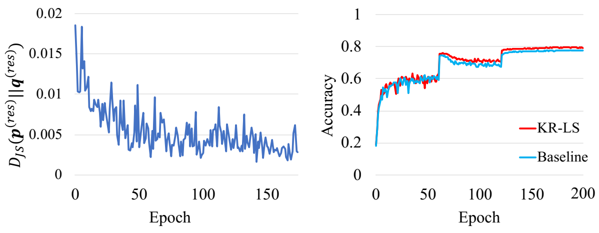


Figure 4: (Left) $D_{JS}(\boldsymbol{p}^{(res)}||\boldsymbol{q}^{(res)})$ on CIFAR-100 dataset. (Right) Test accuracy on CIFAR-100 dataset.

$$\text{where } \mathcal{L}_{smooth} = D_{KL}(\boldsymbol{p}||\boldsymbol{q}^{(smooth)}) \tag{14}$$

In practice, label smoothing enhanced knowledge refinery achieves (KR-LS) better performance consistently. Please refer to Appendix B for the theoretical analysis on label smoothing enhanced knowledge refinery.

Experiments

Knowledge refinery (KR) can be applied to arbitrary neural network. To demonstrate the generalization ability of our method, we perform extensive tests on two general classification tasks: the image classification and the text recognition tasks. Furthermore, we conduct comparison experiments with two well-known soft label-based methods. Finally, we conduct experiments of the sensitivity analysis to demonstrate the robustness of hyper-parameter α. We implemented all experiments with Pytorch framework on a single NVIDIA Tesla V100 GPU.

Image Recognition

Datasets. We use three benchmark datasets of image recognition in our evaluations: (i) CIFAR-10 (Krizhevsky and Hinton 2009): A dataset consisting of $60,000$ images with 32×32 pixel. It has 10 classes, of which each class has $50,000/10,000$ samples in train/test set. (ii) CIFAR-100 (Krizhevsky and Hinton 2009): Similar to CIFAR-10, but it consists of 100 classes, in which each class has 500/100 samples in train/test set. (iii) ImageNet-12 (Russakovsky et al. 2015): A huge image recognition dataset from ILSVRC 2012, which consisting of more than 14 million samples with total $1,000$ classes.

Settings. For CIFAR-10 and CIFAR-100 datasets, we use ResNet-18 (He et al. 2016b) with pre-activation and WideResNet-28-10 (Zagoruyko and Komodakis 2016) as backbone network architectures. We use SGD optimizer with Nesterov momentum (Sutskever et al. 2013) and set an initial learning rate to 0.1, momentum to 0.9 and mini-batch size to 128. The learning rate dropped by 0.1 at the $60/120/160^{th}$ epochs and we train for 200 epochs. The hyper-parameter α in all experiments is an exponential decay function γ^t, where $\gamma = 0.99$. We also adopt data augmentation such as horizontal flips and random crops to samples during the training stage. For ImageNet-12 dataset, we use ResNet-50-v1, ResNet-101-v1,and ResNet-152-v1 (He et al. 2016a) as backbone networks to evaluate our method. In addition, we use

Datasets		CIFAR-10		CIFAR-100	
Model	Method	Error Rate	Params	Error Rate	Params
ResNet18	Baseline	4.72 ± 0.21	11.17M	22.46 ± 0.31	11.22M
	LS	4.59 ± 0.10 ($\downarrow$ 0.13)	11.17M	21.76 ± 0.26 ($\downarrow$ 0.70)	11.22M
	KR*	4.52 ± 0.13 ($\downarrow$ 0.20)	11.17M + 0.1K	21.35 ± 0.25 ($\downarrow$ 1.11)	11.22M + 10K
	KR-LS*	$\mathbf{4.40 \pm 0.14}$ ($\downarrow$ $\mathbf{0.32}$)	11.17M + 0.1K	$\mathbf{20.42 \pm 0.21}$ ($\downarrow$ $\mathbf{2.04}$)	11.22M + 10K
WideResNet-28-10	Baseline	3.87 ± 0.08	36.48M	18.80 ± 0.08	36.54M
	LS	3.72 ± 0.12 ($\downarrow$ 0.15)	36.48M	18.87 ± 0.11 ($\uparrow$ 0.07)	36.54M
	KR*	3.65 ± 0.15 ($\downarrow$ 0.22)	36.48M + 0.1K	18.69 ± 0.17 ($\downarrow$ 0.11)	36.54M + 10K
	KR-LS*	$\mathbf{3.53 \pm 0.25}$ ($\downarrow$ $\mathbf{0.34}$)	36.48M + 0.1K	$\mathbf{18.40 \pm 0.24}$ ($\downarrow$ $\mathbf{0.40}$)	36.54M + 10K

Table 1: Results on CIFAR-10/CIFAR-100. (%) (results averaged over 5 runs, * denotes our method)

Method	ResNet-50		ResNet-101		ResNet-152	
	Top-1 Err.	Top-5 Err.	Top-1 Err.	Top-5 Err.	Top-1 Err.	Top-5 Err.
Baseline	23.87	6.97	22.47	6.18	22.29	6.11
LS	23.75 ($\downarrow$ 0.12)	6.76 ($\downarrow$ 0.21)	22.34 ($\downarrow$ 0.13)	6.07 ($\downarrow$ 0.11)	22.07 ($\downarrow$ 0.22)	5.98 ($\downarrow$ 0.13)
KR*	23.47 ($\downarrow$ 0.40)	6.37 ($\downarrow$ 0.60)	22.09 ($\downarrow$ 0.38)	5.95 ($\downarrow$ 0.23)	21.41 ($\downarrow$ 0.88)	5.56 ($\downarrow$ 0.55)
KR-LS*	**23.20** ($\downarrow$ **0.67**)	**6.22** ($\downarrow$ **0.75**)	**21.98** ($\downarrow$ **0.49**)	**5.81** ($\downarrow$ **0.37**)	**21.23** ($\downarrow$ **1.06**)	**5.29** ($\downarrow$ **0.82**)

Table 2: Results on ImageNet-12. (%) (* denotes our method)

mixed precision training (Micikevicius et al. 2017) in ImageNet experiments which offers significant computational speedup.

Results on CIFAR. The results on two CIFAR datasets are shown in Table 1. In each baseline, we trained the backbone network with only the one-hot encoded label. We compare our proposed knowledge Refinery (KR) and label smoothing enhanced knowledge refinery (KR-LS) with label smoothing (LS) and baselines, of which the best results are marked in bold. As we can see, LS, KR, and KR-LS all outperform baselines of different network structures. Moreover, KR achieves competitive gains over LS, and KR-LS obtains the best performance in all cases consistently, which shows the well complementary between KR and LS. All these observations verify the effectiveness of KR and KR-LS. Specifically, KR-LS achieves 2.04% improvement over the baseline with ResNet-18 model on CIFAR-100 dataset. Note that our KR-based methods improve the performance with only adding negligible parameters to the original neural network. For a K-class task, the extra memory consumption of KR-based methods is $O(K^2)$. Compared with the huge number of parameters of the backbone network, this consumption is exactly negligible. In addition, the training curve is shown in Fig. 4 (Right). We also visualize $D_{JS}(\boldsymbol{p}^{(res)}||\boldsymbol{q}^{(res)})$ in Fig. 4 (Left) to show our KR regularizer make $\boldsymbol{p}^{(res)}$ and $\boldsymbol{q}^{(res)}$ closer during training, which is what we expect.

Results on ImageNet. For the huge ImageNet-12 dataset, the overall results are shown in Table 2, where the Top-N error rate is the fraction of test images for which the ground-truth label is not among the N labels considered most probable by the model. As we can see, compared with the baselines, KR and KR-LS can consistently improve the Top-1 and Top-5 error rate in all cases. Specifically, KR-LS method achieves the best performance which has 1.06% Top-1 and 0.82% Top-5 improvement over those of the baselines in all cases with ResNet-152 model.

Text Classification

Datasets. We use three benchmark datasets of text classification in our evaluations: (i) AGNews (Del Corso, Gulli, and Romani 2005): A topic classification dataset over 4 categories. (ii) Yahoo! Answers[1]: A topic classification dataset over 10 categories. (iii) Yelp Review Full[2]: A dataset of sentiment classification in which polarity star labels ranging from 1 to 5.

Settings. For all datasets, we evaluate knowledge refinery regularizer on four general models for text classification, including FastText (Joulin et al. 2017), TextRNN (Lai et al. 2015), CharCNN (Zhang, Zhao, and Lecun 2015), and Transformer (Vaswani et al. 2017). We compare the baseline and KR-LS in all experiments. The baseline of each model denotes the original network with only hard loss.

Results. The results are shown in Table 3. As we can see, KR-LS can improve the performance under all settings consistently. These experiments exhibit the generalization of our KR method no matter with which backbone network. Therefore, we believe that KR regularizer can be equipped to arbitrary hard label-based methods to improve the performance.

[1]https://webscope.sandbox.yahoo.com

[2]https://www.yelp.com/dataset

Dataset	Method	Baseline	KR-LS
AGNews	FastText	11.03	**10.87** (↓ **0.16**)
	TextRNN	11.15	**10.43** (↓ **0.28**)
	CharCNN	14.22	**12.09** (↓ **2.13**)
	Transformer	10.49	**9.71** (↓ **0.78**)
Yahoo Answer	FastText	29.42	**28.75** (↓ **0.67**)
	TextRNN	31.13	**29.67** (↓ **1.46**)
	CharCNN	29.76	**28.50** (↓ **1.26**)
	Transformer	29.03	**28.76** (↓ **0.27**)
Yelp-Full	FastText	40.66	**40.02** (↓ **0.64**)
	TextRNN	42.81	**41.99** (↓ **0.82**)
	CharCNN	39.28	**39.02** (↓ **0.26**)
	Transformer	38.50	**37.68** (↓ **0.82**)

Table 3: Results of text classification. Metric: Err. (%)

Network	Method	Error Rate	Params.
ResNet-32	Baseline	31.01	$1 \times P_1$
	KD	30.52	$\sim 3 \times P_1$
	DML	29.07	$\sim 2 \times P_1$
	KR-LS	$\mathbf{29.03 \pm 0.37}$	$\sim 1 \times P_1$
WRN-28-10	Baseline	21.31	$1 \times P_2$
	KD	–	–
	DML	19.82	$\sim 2 \times P_2$
	KR-LS	$\mathbf{19.31 \pm 0.18}$	$\sim 1 \times P_2$

Table 4: Results of comparison with DML and KD on CIFAR-100. (%)(results averaged over 5 runs)

Comparison with DML and KD

Settings. We further compare KR-LS with two well-known soft label-based methods, Deep Mutual Learning (DML) (Zhang et al. 2018) and Knowledge Distillation (KD) (Hinton, Vinyals, and Dean 2015). For DML, we use a combination of two sub-network with the same architecture (two ResNet-32 or two WidResNet-28-10 sub-networks) as the backbone network. For KD, we transfer the knowledge from WideResNet-28-10 network to ResNet-32 network. In these experiments, we evaluate all the methods on CIFAR-100 dataset. For fairly comparison, we follow the experimental settings of (Zhang et al. 2018), which set the mini-batch size to 64, and drop the learning rate by 0.1 every 60 epochs. Therefore, the performance of KR-LS here is slightly lower than the results in Table 1.

Results. The results are shown in Table 4, from which we can observe: (i) KR-LS outperforms KD and DML under all settings with the least parameters. (ii) Although KD pre-trains an extra teacher model which provides extra knowledge to the student network, and DML trains two identical model simultaneously with double parameters, KR-LS achieves more performance gains over KD and DML.

Sensitivity Analysis

Settings. In this section, we aim to demonstrate the robustness of the only hyper-parameter in KR (i.e. α in the loss function Eq. 9). Recall that we set α to a exponentially decreasing function $\alpha = \gamma^t$, where t denotes the number of epochs. To evaluate the robustness of α, we set γ to $[0, 0.9, 0.99, 0.995]$ and test the performance of each setting. Note that the setting of $\gamma = 0$ is equivalent to the baseline. In addition, we also set γ to a truncation function as follows:

$$\gamma = \begin{cases} 1 & \text{epochs} < 20, \\ 0 & \text{epochs} \geq 20, \end{cases} \tag{15}$$

, which we called 1^* for simplicity. All settings of γ are shown in Fig. 5 (Right). We conduct all experiments with ResNet-18 model on CIFAR-100 datasets.

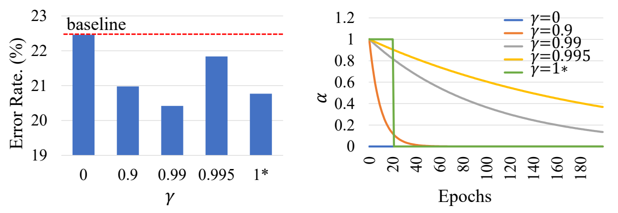


Figure 5: (Left) α with different γ w.r.t. the number of epochs. (Right) the performance w.r.t. γ.

Results. The results are shown in Fig. 5 (Left). As we can see, the performances of $\gamma = [0.999, 0.99, 0.9, 1^*]$ all outperform the performance of baseline. Especially, the setting of $\gamma = 1^*$ almost achieve the best performance with only a slight gap. This result implies that the knowledge refinery regualrizer plays a more role during the initial stage of the training phase and prompts us to explore the mystery behind this phenomenon in the future.

Conclusion

In this paper, we propose Knowledge Refinery (KR), which regularizes DNNs with decoupled labels. The motivation of decoupling the label is the observed softness consistency of the residual label. Extensive experiments on different datasets and different backbone networks consistently show the performance gains over baselines with only negligible memory computation. In the future, we will further investigate the following aspects: (i) exploring how to apply decoupled labels to multi-label tasks, (ii) investigating the theoretical guarantee for KR and decoupled labels.

Acknowledgements

This work is supported in part by the National Key Research and Development Program of China under Grant 2018YFB1800204, the National Natural Science Foundation of China under Grant 61771273, Guangdong Basic and Applied Basic Research Foundation on 2019A1515110344, the China Postdoctoral Science Foundation under Grant 2019M660645, the R&D Program of Shenzhen under Grant JCYJ20180508152204044, the project "PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019)", and the Shenzhen Science and Technology Innovation Commission under Grant JCYJ20180508162601910.

References

Bagherinezhad, H.; Horton, M.; Rastegari, M.; and Farhadi, A. 2018. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641* .

Chen, Z.; Zheng, X.; Shen, H.; Zeng, Z.; Zhou, Y.; and Zhao, R. 2020. Improving Knowledge Distillation via Category Structure. *ECCV* .

Del Corso, G. M.; Gulli, A.; and Romani, F. 2005. Ranking a stream of news. In *Proceedings of the 14th international conference on World Wide Web*, 97–106. ACM.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *CVPR*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *international conference on machine learning* 448–456.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of Tricks for Efficient Text Classification. *conference of the european chapter of the association for computational linguistics* 2: 427–431.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lai, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Lee, H. B.; Lee, J.; Kim, S.; Yang, E.; and Hwang, S. J. 2018. DropMax: Adaptive variational softmax. In *Advances in Neural Information Processing Systems*, 919–929.

Lopez-Paz, D.; Bottou, L.; Schölkopf, B.; and Vapnik, V. 2015. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643* .

Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740* .

Müller, R.; Kornblith, S.; and Hinton, G. E. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 4694–4703.

Pereyra, G.; Tucker, G.; Chorowski, J.; Kaiser, Ł.; and Hinton, G. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548* .

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115(3): 211–252.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1): 1929–1958.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* .

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, 1139–1147.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*, 2818–2826.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. *neural information processing systems* 5998–6008.

Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *International conference on machine learning*, 1058–1066.

Wang, Z.; Liu, L.; and Tao, D. 2020. Deep streaming label learning. In *International Conference on Machine Learning*, 9963–9972. PMLR.

Xie, L.; Wang, J.; Wei, Z.; Wang, M.; and Tian, Q. 2016. Disturblabel: Regularizing cnn on the loss layer. In *CVPR*, 4753–4762.

Xu, K.; Rui, L.; Li, Y.; and Gu, L. 2020. Feature Normalized Knowledge Distillation for Image Classification. *ECCV* .

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* .

Zhang, X.; Zhao, J. J.; and Lecun, Y. 2015. Character-level convolutional networks for text classification. *neural information processing systems* 649–657.

Zhang, Y.; Xiang, T.; Hospedales, T. M.; and Lu, H. 2018. Deep mutual learning. In *CVPR*, 4320–4328.

Zhu, X.; Gong, S.; et al. 2018. Knowledge Distillation by On-the-Fly Native Ensemble. In *Advances in Neural Information Processing Systems*, 7517–7527.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *CVPR*, 8697–8710.