Temporal Calibrated Regularization for Robust Noisy Label Learning

Dongxian Wu[1,3] Yisen Wang[2] Zhuobin Zheng[1] Shu-Tao Xia[1,3] †
[1]Tsinghua University [2]Shanghai Jiao Tong University
[3]PCL Research Center of Networks and Communications, Peng Cheng Laboratory

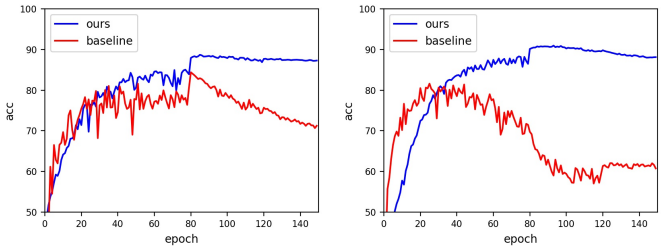
***Abstract*—Deep neural networks (DNNs) exhibit great success on many tasks with the help of large-scale well annotated datasets. However, labeling large-scale data can be very costly and error-prone so that it is difficult to guarantee the annotation quality (*i.e.*, having noisy labels). Training on these noisy labeled datasets may adversely deteriorate their generalization performance. Existing methods either rely on complex training stage division or bring too much computation for marginal performance improvement. In this paper, we propose a Temporal Calibrated Regularization (TCR), in which we utilize the original labels and the predictions in the previous epoch together to make DNN inherit the simple pattern it has learned with little overhead. We conduct extensive experiments on various neural network architectures and datasets, and find that it consistently enhances the robustness of DNNs to label noise.**

I. Introduction

Deep Neural Networks (DNNs) have demonstrated extraordinary performance in solving many complex problems, including computer vision [1], natural language processing [2] and speech recognition [3]. The success of DNNs heavily relies on the large-scale well-annotated datasets. However, well-annotated datasets are not always available [4]–[6]. Instead, datasets that contain incorrectly-annotated labels (*i.e.*, noisy labels) are very common. Unfortunately, DNNs are susceptible to such noisy labels, easily raising the problem of overfitting [7]. Training accurate CNNs against noisy labels is therefore of great practical importance.

Recent studies have tried to tackle this issue by dividing the DNN training procedure into two stages, *i.e.*, early stage pattern learning and later stage label memorization, either by manual setting [8] or a complex scheduler [9], [10]. However, DNNs may start overfiting at any time. For example, as shown in Fig. 1, a PreAct-ResNet-18 [11] starts overfitting around the 30-*th* epoch (150 epochs in total), while a ResNet-32 [12] starts overfitting after 80-*th* epoch. Such methods are limited in practice, since it is hard to predict which epoch start overfitting. Gradient correction [13], [14] is another kind of method. However, it is directly based on the given labels, even when there are noisy labels, thus updating parameters with these noisy gradients will make DNNs easier to overfit. Therefore, there are a certain of work modifying the gradients by the aid of meta-learning [13], [15] or bilevel optimization [14], which require not only extra clean or synthetic noisy samples, but also more forward and backward passes.

†Corresponding author: Shu-Tao Xia (xiast@sz.tsinghua.edu.cn)



(a) ResNet-32 (b) PreAct-ResNet-18

Fig. 1. We trained different architectures in CIFAR-10 with 40% uniform noise. Although DNN has two stages in learning, the division time relies on its specific setting, which makes two stages method hard to generalize.

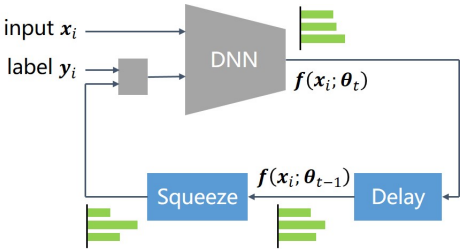


Fig. 2. The framework of the proposed Temporal Calibrated Regularization, which consists of a temporal delay (for the reflection loss) and a squeeze function. The network is optimized using the combination of original labels and predictions processed by TCR.

In this paper, we propose an extremely simple but effective label correction method, called Temporal Calibrated Regularization (TCR), as shown in Fig. 2. Specifically, we use a convex combination of raw labels and the DNN predictions of the previous epoch as the pseudo-labels for training targets. Besides, we propose to utilize a "Squeeze" function, which encourages DNN to have a little higher confidence in its predictions. In TCR, the prediction in the previous epoch is the target for DNN to inherit the simple pattern learned by itself. In this case, DNN can adapt fast without manual stage division or complex scheduler design as we only use the recent historical information. Our method is therefore easy to implement and extend to different situations.

Our main contributions are:

- We propose a simple but effective method, Temporal Calibrated Regularization, for robust learning, which considers both the original raw labels and DNN historical predictions so as to inherit the simple pattern learned

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

before and hinder overfitting to noisy labels.
- To counteract the adverse effects of resistance degradation, we introduce "Squeeze" technique to tackle this problem by amplifying the largest prediction to encourage the model to have both distinguishable and high confidence in predicting labels.
- We conduct extensive experiments on various network architectures and datasets, and demonstrate the competitive performance of the proposed method in image classification tasks compared to state-of-the-art methods..

II. Related Work

Learning with noisy labels has been widely investigated in [16], and a variety of approaches have been proposed to robustly train DNNs on noisy datasets, such as regularization and correction. Since [17] showed that dropout can hinder overfitting, other effective regularizations, e.g. mixup [18] and Bilevel Optimization [14], were developed. Another approach is to use label correction to eliminate the influence of noisy labels. Reference [19] replaced the target labels with a convex combination of original labels and current predictions from DNN. Further, A complex combination strategy was utilized based on local intrinsic dimensionality in [9]. Joint optimization was proposed in [8] to jointly optimize network parameters and data labels during training. Moreover, there are approaches to formulate noise models explicitly or implicitly through DNNs [20]–[23], conditional random field [24] and knowledge graphs [25]–[27]. These noise models are then applied to correct the loss, infer the true labels or assign smaller weights to noisy samples. Other studies introduced robust losses [28]–[33] through noise modeling. However, these methods rely on the modeling assumptions and are limited to generalize to complex situations. They may also request extra clean data or use expensive estimation methods.

Our proposed Temporal Calibrated Regularization (TCR) aims at keeping the simple pattern that DNN learned before and impede "brutally" memorizing noisy labels through a label correction method. It also can be regarded as a simplified modifying gradient method.

III. Temporal Calibrated Regularization

In this section, we first provide the preliminaries and notations, then followed by the definition of the proposed reflection loss, gradient analysis and resistance degradation in our proposed Temporal Calibrated Regularization (TCR).

A. Preliminaries and Notations

Column vectors and matrices are denoted in bold (e.g. $\boldsymbol{x}$). $\mathbf{1}$ is a vector of all ones and $\boldsymbol{e}^i$ denoted the ith standard canonical vector in $\mathbb{R}^c$, i.e. $\boldsymbol{e}^i \in \{0,1\}^c, \mathbf{1}^\top \boldsymbol{e}^i = 1$. In a typical c-class classification problem, we are given a set of n training inputs $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$ with their corresponding labels $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n\}$ within label space $\mathcal{Y} = \{\boldsymbol{y} : \boldsymbol{y} \in [0,1]^c, \mathbf{1}^\top \boldsymbol{y} = 1\}$. Given an input $\boldsymbol{x}_i$, the output vector of the DNN with parameter $\boldsymbol{\theta}$ is $\boldsymbol{h}(\boldsymbol{x}_i;\boldsymbol{\theta})$. After a softmax function, the probability vector of prediction is $\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta})$, i.e. $\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}) = \text{softmax}(\boldsymbol{h}(\boldsymbol{x}_i;\boldsymbol{\theta}))$. The objective function is an empirical risk, such as the cross entropy loss, as follows:

$$L(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i^\top \log \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}). \tag{1}$$

In a clean training dataset, the parameters $\boldsymbol{\theta}$ of a DNN is optimized by minimizing (1) using gradient descent method. The derivative over the DNN output $\boldsymbol{h}$ is

$$\frac{\partial L}{\partial \boldsymbol{h}} = -\sum_{i=1}^{n} \big(\boldsymbol{y}_i - \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta})\big). \tag{2}$$

With the chain rule, we can update the parameters as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \epsilon \nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \epsilon \Big(\frac{\partial L}{\partial \boldsymbol{h}}\Big)^\top \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta}_t}. \tag{3}$$

In this work, we consider a classification problem with noisy labels. Let $\tilde{\boldsymbol{y}}_i$ be the noisy label for input $\boldsymbol{x}_i$, and we are only given these noisy labels $\{\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{y}}_2, \cdots, \tilde{\boldsymbol{y}}_n\}$. The connection between clean labels and noisy labels is based on the transition probability,

$$p(\tilde{\boldsymbol{y}}|\boldsymbol{x}) = \sum_{\boldsymbol{y}} p(\tilde{\boldsymbol{y}}|\boldsymbol{y},\boldsymbol{x}) p(\boldsymbol{y}|\boldsymbol{x}). \tag{4}$$

Assuming noise is conditionally independent of inputs [20], given the true labels so that

$$p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^k | \boldsymbol{y} = \boldsymbol{e}^j, \boldsymbol{x}) = p(\tilde{\boldsymbol{y}} = \boldsymbol{e}^k | \boldsymbol{y} = \boldsymbol{e}^j) = T_{jk}. \tag{5}$$

In general, this noise is defined to be class dependent. Noise is uniform with noise rate η, if

$$\begin{aligned} T_{jk} &= 1-\eta, \text{if } j = k \\ T_{jk} &= \frac{\eta}{c-1}, \text{if } j \neq k. \end{aligned} \tag{6}$$

B. Reflection Loss

As discussion in previous studies [9], [17], there are two stages in training procedure, including a early stage of simple pattern learning and a late stage of label memorization. Our work aims at motivating DNNs to take into consideration not only noisy labels, but also simple patterns learned before, which mitigates overfitting to noisy labels. Specifically, we utilize both the original labels $\tilde{\boldsymbol{y}}_i$ and the predictions $\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1})$ in the previous epoch as the training target together. The resulting loss (*e.g.,* cross entropy loss) becomes

$$\begin{aligned} &L(\boldsymbol{\theta}_t; \boldsymbol{\theta}_{t-1}) \\ =& -\frac{1}{n}\sum_{i=1}^{n} \big(\beta \tilde{\boldsymbol{y}}_i + (1-\beta)\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1})\big)^\top \log \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t) \\ =& \beta L_o + (1-\beta) L_r, \end{aligned} \tag{7}$$

where the former L_o is the original loss in clean setting and the latter L_r provides resistance to label noise. The weighted coefficient, β, is the "confidence" of annotation quality. If we are more confident in the label quality (*i.e.,* larger β), DNN relies more on original raw labels. Otherwise, DNN refines more from its last predictions, corresponding to its learned patterns.

Since we utilize a convex combination of labels in the dataset and predictions in the previous epoch, our reflection loss belongs to pseudo-label method [8], [9], [19]. After separating our loss in (7), the second part can be regarded as a special consistency loss, in which we replace mean squared error (MSE) with cross entropy (CE) and utilize predictions only in the previous epoch instead of ensembling historical predictions.

C. Gradient Analysis

In this part, we explore the mechanism behind our proposed loss and discuss its merits. In conventional gradient descent algorithm, gradients calculated from noisy samples may lead to overfitting [13] as shown in Fig. 3. Notably, our method provides a resistance to these noisy gradients, which is apparent after derivation of the gradient over the DNN outputs $\boldsymbol{h}$ for a given sample $\boldsymbol{x}$:

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{h}} = &-(1-\beta)\sum_i \big(\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}) - \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t)\big) \\ &-\beta\sum_i \big(\tilde{\boldsymbol{y}}_i - \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t)\big).\end{aligned} \tag{8}$$

Other correction methods, such as bootstrap method [19], have similar decomposition and have different resistance mechanism from ours. Given a sample $\boldsymbol{x}$, for bootstrap-soft method, we derivate the gradients of the pesudo-label term over the output h_i (before softmax) of DNN for class l:

$$\frac{\partial L}{\partial h_l} = f_l(\sum_j f_j \log f_j - \log f_l), \tag{9}$$

where f_l is the output probability (after softmax) of class l, *i.e.* $f_l^t = f_l(\boldsymbol{x};\boldsymbol{\theta}_t)$.

For boostrap-hard, the result is

$$\frac{\partial L}{\partial h_l} = f_l^t - \mathbb{I}(l = \arg\max_j f_j^t). \tag{10}$$

Assuming that we have a sample with prediction of class k, that is $f_k \geq f_l$, the gradients for different class satisfying

$$\frac{\partial L}{\partial h_k} \geq \frac{\partial L}{\partial h_l}, l = 1, 2, \cdots, c \tag{11}$$

which means the pesudo-label term push the predictions more confident.

For our reflection loss, the gradients is

$$\frac{\partial L}{\partial h_l} = f_l^t - f_l^{t-1}. \tag{12}$$

Obviously, Equation (12) achieves satisfying performance without pushing the prediction more confidence, which makes it is suitable for open-set noise [34].

D. Squeeze Technique

Note that the resistance in reflection loss will degrade as the learning rate lr decays, which is also called the *Resistance Degradation Problem*. For a given sample $\boldsymbol{x}_i$, the resistance in (8) is $\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}) - \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t)$, which is related to the step size. Even with same gradients to descend, smaller step size results in smaller difference $\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}) - \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\theta}_t)$, which degrades the resistance in next iteration. As a result, the resistance is no longer enough to overcome overfitting.

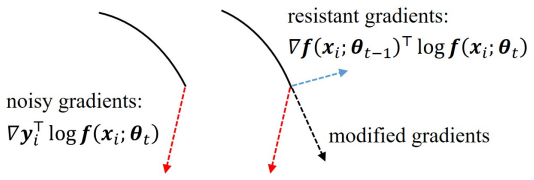


Fig. 3. The illustration of gradient correction. Left: gradients from noisy samples may lead to overfit; Right: noisy gradients is modified by correction gradients, which is less prone to overfit to noisy labels.

Thus, we propose a technique called "Squeeze" to keep enough resistance after learning rate decays. Since the DNN has learned some simple pattern, we could make DNN be more confident in its predictions. Assuming the probability for a sample is $\boldsymbol{p}$, the squeeze function is

$$\text{Squeeze}(\boldsymbol{p}) = \frac{\boldsymbol{p}^\gamma}{\boldsymbol{1}^\top \boldsymbol{p}^\gamma}, \tag{13}$$

where $(\cdot)^\gamma$ is the element-wise power to $\gamma \geq 1$. The squeeze function becomes one-hot function as $\gamma \to \infty$. With this function, largest value in prediction vector becomes a bit larger, so as to provide larger gradient for the most probable class and smaller ones for potential noisy samples.

Compared with other techniques to improve DNN confidence (*e.g.*, minimum entropy regularization in [8]), we only modify labels and keeps others unchanged. Besides, entropy regularization is severely affected by the number of classes, which makes hyper-parameter settings hard to extend, e.g. from CIFAR-10 to CIFAR-100, while our method could remain the same hyper-parameters.

E. The Full Algorithm of TCR

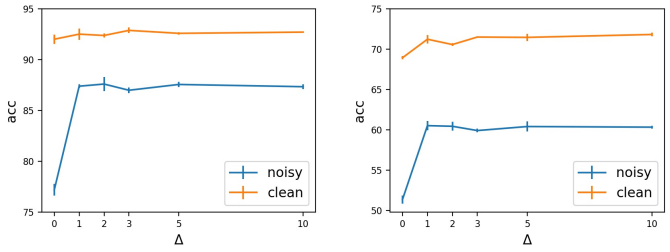
The detailed procedure is outlined in Algorithm 1. Before updating the parameters, we obtain the current predictions of the DNN (Line 5). After the update, we record these prediction in the storage z. The pseudo-labels for loss is the convex combination of original labels and predictions recorded in the storage, *i.e.*, the prediction in the previous epoch (Line 6-7). Since there is no last predictions for the first epoch, we use the original labels in first epoch (Line 8-9). To improve the DNN confidence over its predictions and compensate the degraded gradients after learning rate decays, we utilize the "Squeeze" function. TCR owns the following good properties:

Easy Implementation Note that, TCR is a general extension and easy to implement requiring few modifications. All we need is to simply add two parts, a Temporal Delay and a Squeeze technique, in conventional models as shown in Fig. 2. The former stores predictions in current epoch and provides its collections for DNN in next epoch.

Less Time and Computation Cost Compared with the prior work [13]–[15] in gradient correction under label noise, our method is just a label correction method based on prediction

Algorithm 1: Temporal Calibrated Regularization

Randomly initialize $\boldsymbol{\theta}$;
$Z = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_n] \leftarrow \mathbf{0}_{[c \times n]}$;
for t *in* $[1, num\_epochs]$ **do**
  for *each mini-batch* B **do**
    $\boldsymbol{z}_i^{(t)} \leftarrow \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta})$;
    if $t \geq 1$ **then**
      $\boldsymbol{y}_i^* \leftarrow \beta \boldsymbol{y}_i + (1-\beta)\boldsymbol{z}_i$
    else
      $\boldsymbol{y}_i^* \leftarrow \boldsymbol{y}_i$
    end
    $L \leftarrow \sum_{i \in B} \boldsymbol{y}_i^{*\top} \log \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta})$;
    update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_t - \epsilon \nabla_{\boldsymbol{\theta}} L$;
    $\boldsymbol{z}_i \leftarrow \boldsymbol{z}_i^{(t)}$;
    if $t \geq T_s$ **then**
      $\boldsymbol{z}_i \leftarrow \text{Squeeze}(\boldsymbol{z}_i)$
    end
  end
end
return $\boldsymbol{\theta}$



(a) a (b) b

Fig. 4. Accuracy(%) on clean and noisy (40% uniform noise) CIFAR-10 and CIFAR-100 with variable time difference Δ. As long as we use the predictions before, we achieve benefits. And the time difference has negligible effects.

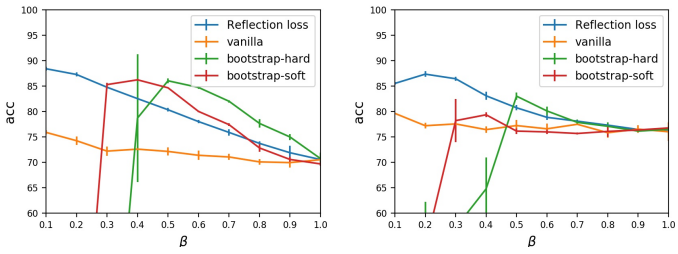
difference between adjacent epochs, and has less overhead. The only storage overhead in our method is the memory for predictions of DNN in the previous epoch, which is proportional to $O(c \cdot n)$. The time overhead in our method is almost negligible.

IV. Experiments

In this section, we firstly conduct a series of comparative experiments to empirically understanding our proposed method. We then evaluate the robustness of our proposed model to noisy labels with comprehensive experiments on different noise types and different noise ratio.

A. Understanding the Learning Process

Recall that we use the predictions in the previous epoch, we test whether to obtain more from predictions of earlier times. We test the results of ResNet-32 in clean and noisy (40% uniform noise) CIFAR-10 and CIFAR-100 with varying time difference $\Delta = \{0, 1, 2, 3, 5, 10\}$, *i.e.*, in the current epoch t, we utilize the prediction in the epoch $t - \Delta$. As shown in Fig. 4, there is negligible difference between different Δ, except the case $\Delta = 0$, which actually uses current predictions. The results indicate that the determining factor is to let DNN consider its previous predictions, and how long does not matter. From a perspective of implementation, the space complexity is $O(n \cdot \Delta)$, large Δ takes more challenge in storage. In a word, $\Delta = 1$ has the satisfying performance and lowest space complexity.



(a) 40% uniform noise (b) 40% asymmetric noise

Fig. 5. Accuracy of different label correction methods on noisy CIFAR-10 with varying hyper-parameter β. The vanilla method just reduce the learning rate and has little effect. The bootstrap-hard and bootstrap-soft are sufficient but fragile as β is small. Our reflection loss is stable in different choices of β.

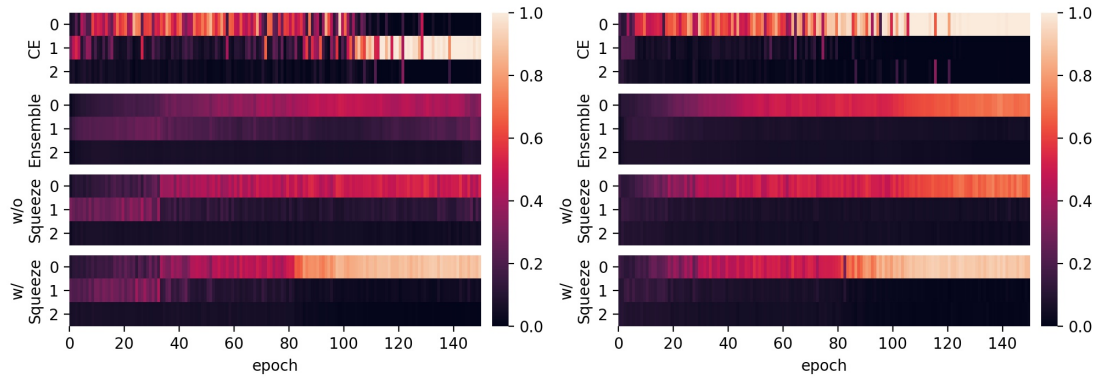
B. Comparison to Label Correction Methods

Experimental setup: We utilize the PreAct-ResNet18. The networks are trained on 40% uniform noise. All experiments use the mini-batch of size 128 and networks were trained using SGD with momentum 0.9, weight decay 10^{-4} and an initial learning rate of 0.1. The learning rate is divided by 10 after epochs 80, 120 (150 epochs in total). As we have discussed the difference to correction methods, we now empirically compare their performance to show the advantage of TCR in CIFAR-10 with 40% uniform noise or 40% asymmetric noise. Since different methods may have different choices of hyper-parameter β, we test the performance in $\beta = \{1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$.

Results: Fig. 5 demonstrates the the influence of different correction methods. We can observe that the vanilla method which uses the current predictions has a slightly better results. The correction methods like bootstrap-soft and bootstrap-hard have satisfying performance only in specific β, since they have to keep balance between enough resistance and pattern learning in the beginning. For our method, it can learn pattern and keep enough resistance in smaller β as discussed before.

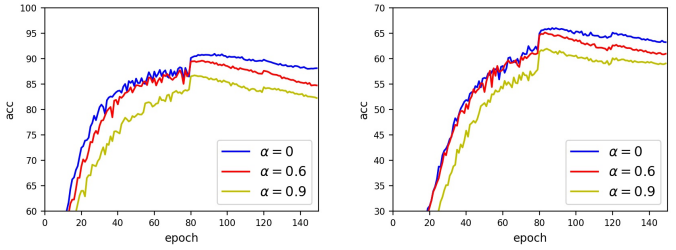
C. Comparison to Temporal Ensemble

Although our reflection loss can be regarded as a special case of the consistency loss in Temporal Ensemble [10], in which we replace the Mean Square Error with Cross Entropy and only utilize predictions in the previous epoch. We empirically demonstrate that the latter is a significant setting.



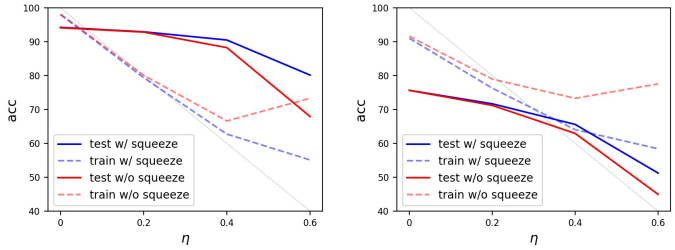
(a) an "airplane"(row 0) with a label "automobile"(row 1) (b) an "airplane"(row 0) with a correct label

Fig. 6. We visualize the predicted probability of DNN during training for different losses. From the top to bottom, they cross entropy loss, reflection loss with EWA, the original reflection loss and our method. The conventional approach with CE learns simple pattern first but memorizes the noisy label at last. while our methods learn simple pattern and keeps it until the end.



(a) CIFAR-10 (b) CIFAR-100

Fig. 7. Accuracy in clean test set during training on CIFAR-10 with 40% uniform noise. If we set α a large value (0.9), it hampers learning simple pattern in early training since the poor predictions early influence longer. If we set $\alpha = 0.6$, it has learned simple pattern in early stage, but start overfitting severely after learning rate decays.



(a) CIFAR-10 (b) CIFAR-100

Fig. 8. Accuracy(%) on training set and test set of CIFAR-10 with varying ratio of uniform noise. As the noise ratio increase, DNN without "Squeeze" overfits to noisy labels severely (red dashed line), and the test accuracy decreases (red solid line). "Squeeze" impedes this problem effectively as shown in blue line.

The original Temporal Ensemble utilizes the Exponentially Weighted Average (EWA) to average predictions as*

$$(1-\alpha)\boldsymbol{f}_i^{(t)} + (1-\alpha)\alpha\boldsymbol{f}_i^{(t-1)}(1-\alpha)\alpha^2\boldsymbol{f}_i^{(t-2)} + \cdots, \quad (14)$$

while we find it harms the learning from two aspects: 1) slowing down the learning process in early stage; 2) making DNN easier to overfitting. To validate this, we utilize the same setting in Section 4.2 except for replacing the predictions in the previous epoch with the ensemble predictions and set $\alpha = \{0, 0.6, 0.9\}$ respectively. The accuracy in test set is plotted in Fig. 7.

If we set α a large value (0.9), it hampers learning simple pattern in early training since the poor predictions early influence longer. If we set $\alpha = 0.6$, it has learned simple pattern in early stage, but start overfitting severely after learning rate decays. As shown in Fig. 6, the predicted probabilities for model with $\alpha = 0.6$ are more smoothing, while ours are fluctuate, which have larger difference in temporal dimension. According to (8), the resistant gradient is related to the difference in temporal dimension. Thus, $\alpha = 0$ adds more noise in gradients comparing to $\alpha = 0.6$, which improves its robustness.

*Here, we denote $\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}_t)$ as $\boldsymbol{f}_i^{(t)}$ for simplicity.

D. The Effect of Squeeze Technique

To validate the significance of the squeeze function in our method, we conduct a series of experiments. First, we visualize the predicted probability for samples with a correct or incorrect label in Fig. 6. For a DNN only with our reflection loss (no squeeze), its prediction confidence is not high enough whether the label is correct or not. As introducing the squeeze function after the first learning rate decay, its prediction confidence becomes higher. Next, we plot the accuracy in noisy train set and clean test set during training, as shown in Fig. 8. In severe uniform noise, the reflection loss encounters memorizing noisy samples (accuracy in noisy train set is larger than the ratio of correct samples), and the squeeze technique overcomes this problem significantly.

E. Robustness against Noisy Labels

Finally, we evaluate the robustness of our method against noisy labels under different conditions (various noise types and ratios, architectures, etc.) to demonstrate the advantageous performance of our method.

TABLE I

AVERAGE ACCURACY (%) IN CLEAN TEST SET (5 RUNS) WITH VARYING NOISE RATIOS IN UNIFORM NOISE AND ASYMMETRIC NOISE. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**. SINCE FORWARD LOSS REQUIRES THE GROUND TRUTH CONFUSION MATRIX, WE ALSO HIGHLIGHT THE SECOND ONE IN **BOLD** IF THE RESULT OF FORWARD LOSS IS THE HIGHEST ONE.

Datasets	Method	clean	Symmetric Noise			Asymmetric Noise			
			0.2	0.4	0.6	0.1	0.2	0.3	0.4
CIFAR-10	CE	94.31	80.13	60.75	39.70	90.84	86.19	81.98	76.83
	Forward	94.31	86.37	76.52	60.49	92.55	90.06	88.67	86.06
	GCCE($q = 0.7$)	93.35	91.51	87.18	68.04	92.04	89.75	82.78	76.32
	mixup($\alpha = 8$)	**94.46**	92.17	88.55	80.14	93.77	92.56	90.94	86.61
	Joint	91.49	90.34	89.41	79.68	92.45	92.28	91.32	90.30
	Ours w/o Squeeze	94.16	92.84	88.25	67.93	93.60	93.09	92.12	89.11
	Ours	94.24	**92.92**	**90.50**	**80.15**	**93.90**	**93.16**	**92.52**	**90.53**
CIFAR-100	CE	74.30	59.38	43.71	24.53	68.35	61.25	53.91	44.42
	Forward	74.50	63.94	51.66	37.66	73.45	73.02	**72.18**	**71.92**
	GCCE($q = 0.7$)	69.76	67.32	64.08	50.70	68.58	66.75	64.22	51.84
	mixup($\alpha = 8$)	74.33	68.68	58.56	42.78	71.44	67.45	61.12	50.06
	Ours w/o Squeeze	75.64	71.21	62.94	44.95	74.52	72.65	67.73	56.06
	Ours	**75.65**	**71.68**	**65.59**	**51.23**	**74.95**	**73.81**	**71.19**	**63.15**

Experimental setup: Experiments were conducted on widely used datasets: CIFAR-10 and CIFAR-100 with different architectures like ResNet34, ResNet44, PreAct ResNet-18 and Wide ResNet (WRN). All Networks are trained using SGD with momentum 0.9, weight decay 10^{-4} and an initial learning rate of 0.1. The learning rate is divided by 10 after epochs 80, 120 (150 epochs in total) except for Wide ResNet. We train WRN with mini-batch size 100, and divide its learning rate after epochs 80, 100 (120 epochs in total), which is same as the description in [15]. Simple data augmentations (width/height shift and horizontal flip) are applied. For our method, TCR, we all set $\beta = 0.1, \gamma = 1.1$ in each experiment of this section.

Baselines: We compare our method with state-of-the-art baselines for noisy label learning. The baselines include: 1) Cross Entropy Loss: the conventional approach of training with cross-entropy loss; 2) Forward Loss [20]: a robust loss through multiplying the prediction of DNN by a estimated label transition probability matrix; 3) Generalized Cross Entropy Loss (GCCE) [32]: a noise-robust loss with a hyper-parameter q, which can be seen as a generalization of cross-entropy loss ($q = 0$) and mean absolute error ($q = 1$). Here, $q = 0.7$ is used as suggested in [32]; 4) mixup [18]: a special data augmentation using a linear combination of images and labels. We tested $\alpha = \{1, 4, 8, 32\}$, and set $\alpha = 8$ which has the best results; 5) Joint [8]: a joint optimization framework of learning DNN parameters and estimating true labels. Here, we use the first part of Joint since other methods are only trained once. The parameters are set according to the original paper, $lr = 0.08, \alpha = 1.2, \beta = 0.8$ for symmetric noise, $lr = 0.03, \alpha = 0.8, \beta = 0.4$ for asymmetric noise.

Robustness to Partially Corrupted Labels: Asymmetric noise and uniform (symmetric) noise are typical settings of partially corrupted labels. Both noises are generated according to [20]. All experiments are conducted with PreAct ResNet-18. We report the average accuracy over 5 repetitions of the experiments in Table I. For noisy CIFAR-10, our proposed method outperforms the baselines significantly. Actually, the "reflection" loss already has a comparative performance to mixup. However, our method is little worse in clean training data, which we conjecture is because they resist to memorize all training samples and the training accuracy cannot be close to 100% in only 150 epochs. For noisy CIFAR-100, our method has the best performance overall. Note that Forward sometimes also delivers a relatively good performance, as we directly provide it with the ground truth noise matrix which is not often available in real-world settings. In conclusion, our method has advantageous performance and generality for different noise types and ratios.

Extension to Different Architectures: For more comprehensive comparison with other methods proposed recently and validation of the generality of our method, we conduct experiments with different architectures, including ResNet-44 [12], ResNet-34 [12], PreAct ResNet-18 [11] and WRN [35]. We also list the reported accuracy from other state-of-art methods, such as D2L [9], GCCE [32], Bilevel [14], mixup [18] (our reproduced implementation) and a MAML method [15]. The results are shown in Table II. Our method has consistently robustness with all architectures. In every architectures, ours outperform its corresponding competitors. For different architectures, we all set the hyper-parameters $\beta = 0.1, \gamma = 1.1$, which indicates our method is easy to extend.

Performance on Open-set Noise: Open-set noise [34] is another common label noise in real data, in which a noisy sample possesses a true class that is not contained within the set of known classes. When we collect data through search engines, it is impossible to ensure the noisy data only from the known classes we have interests in. We use the open-set noisy dataset, CIFAR-10 + CIFAR-100 as described in [34]. The result is shown in Table III and our method outperforms its competitors. In open-set noise, it is not a good choice to push predictions to be more confident, as some images do not belong to any class. This may be the reason that bootstrap-hard (boots in Table III) has worsen performance than ours.

Experiments on Real-World Data with Noisy Labels: Lastly, we test our method on the real-world large-scale noisy dataset Clothing1M [21], which consists of fashion images belonging to 14 classes: T-shirt, Shirt, Knitwear, Chif-

TABLE II
ACCURACY (%) IN CLEAN TEST SET OF DIFFERENT ARCHITECTURES WITH 40% UNIFORM NOISE. WE ALSO LIST THE REPORTED ACCURACY FROM OTHER STATE-OF-ART METHODS.

Method	Architecture	CIFAR-10	CIFAR-100
CE	ResNet-44	66.17	45.71
	ResNet-34	65.45	45.30
	PreAct ResNet-18	60.75	43.71
	WRN	78.28	50.40
D2L	ResNet-44	-	52.01
GCCE	ResNet-34	87.13	61.77
Bilevel	PreAct ResNet-18	87.0	59.8
mixup	PreAct ResNet-18	88.55	58.56
MAML	WRN	86.92	61.34
Ours	ResNet-44	89.60	58.13
	ResNet-34	90.72	64.58
	PreAct ResNet-18	90.50	65.59
	WRN	91.43	67.35

TABLE III
ACCURACY (%) IN CLEAN TEST SET WITH 40% OPEN-SET NOISE.

noise type	CE	GCCE	boots	mixup	ours
CIFAR100	86.73	87.52	88.26	91.97	92.29

fon, Sweater, Hoodie, Windbreaker, Jacket, Down Coat, Suit, Shawl, Dress, Vest, and Underwear. The labels are generated by the surrounding text of images and are thus extremely noisy. The overall accuracy of the labels is $\sim 61.54\%$, with some pairs of classes frequently confused with each other (*e.g.*, Knitwear and Sweater), which may contain both symmetric and asymmetric label noise. We used ResNet-50 pre-trained on ImageNet to align experimental condition with [20]. For preprocessing, we resize the image to 256×256, crop the middle 224×224 as input, and perform normalization. We use a batch size $k = 32$, learning rate $lr = 0.0008$, and update using SGD with momentum 0.9 and weight decay 10^{-3}.

We utilize 3 epochs to train DNN on noisy data, 1 epoch for normal training, 1 epoch for only reflection loss, and 1 epoch for the full algorithm. We still set $\beta = 0.1, \gamma = 1.1$. We achieve 72.54% at last, which is better than 68.94% from Cross Entropy Loss (quoted from [20]), 69.8% from Forward Loss [20], 69.9% from Bilevel [14], and 72.16% from [8].

V. CONCLUSION

In this paper, we proposed a label-correction method, called Temporal Calibrated Regularization (TCR), to let DNN consider the simple pattern it has learned and labels in dataset simultaneously, so as to overcome the overfitting to noisy labels. Specifically, TCR utilizes the convex combination of original labels and predictions in the previous epoch as pseudo-labels and applies "Squeeze" to improve the DNN confidence for its predictions. Experiments on various network architectures and datasets demonstrated that our method significantly outperforms the state-of-the-arts.

VI. ACKNOWLEDGMENT

Shu-Tao Xia is supported in part by National Key Research and Development Program of China under Grant 2018YFB1800204, National Natural Science Foundation of China under Grant 61771273, R&D Program of Shenzhen under Grant JCYJ20180508152204044, and research fund of PCL Future Regional Network Facilities for Large-scale Experiments and Applications (PCL2018KP001).

TABLE IV
TEST ACCURACY (%) ON THE CLOTHING1M DATASET

Method	CE	Forward	Bilevel	Joint	Ours
Acc	68.94	69.84	69.9	72.16	72.54

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.
[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.
[3] Y. Wang, X. Deng, S. Pu, and Z. Huang, "Residual convolutional ctc networks for automatic speech recognition," *arXiv preprint arXiv:1702.07793*, 2017.
[4] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.
[5] R. Tanno, A. Saeedi, S. Sankaranarayanan, D. C. Alexander, and N. Silberman, "Learning from noisy labels by regularized estimation of annotator confusion," in *CVPR*, 2019.
[6] J. Wu, S. Pan, X. Zhu, C. Zhang, and X. Wu, "Multi-instance learning with discriminative bag mapping," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 6, pp. 1065–1080, 2018.
[7] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR*, 2017.
[8] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *CVPR*, 2018.
[9] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *ICLR*, 2018.
[10] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *ICLR*, 2017.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.
[12] ——, "Deep residual learning for image recognition," in *CVPR*, 2016.
[13] J. Li, Y. Wong, Q. Zhao, and M. Kankanhalli, "Learning to learn from noisy labeled data," *arXiv preprint arXiv:1812.05214*, 2018.
[14] S. Jenni and P. Favaro, "Deep bilevel learning," in *ECCV*, 2018.
[15] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *ICML*, 2018.
[16] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *TNNLS*, vol. 25, no. 5, pp. 845–869, 2014.
[17] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *ICML*, 2017.
[18] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2017.
[19] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *ICLR*, 2015.
[20] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *CVPR*, 2017.
[21] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *CVPR*, 2015.
[22] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *ICLR*, 2015.
[23] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *ICLR*, 2016.
[24] A. Vahdat, "Toward robustness against label noise in training deep discriminative neural networks," in *NeurIPS*, 2017.
[25] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, "Learning from noisy labels with distillation," in *ICCV*, 2017.

[26] W.-X. Lu, C. Zhou, and J. Wu, "Big social network influence maximization via recursively estimating influence spread," *Knowledge-Based Systems*, vol. 113, pp. 143–154, 2016.
[27] J.-X. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li, "Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection," in *CVPR*, 2019.
[28] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.
[29] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.
[30] B. Van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," in *NeurIPS*, 2015.
[31] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *NeurIPS*, 2013.
[32] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *NeurIPS*, 2018.
[33] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *ICCV*, 2019.
[34] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, "Iterative learning with open-set noisy labels," in *CVPR*, 2018.
[35] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.