PENCIL: Deep Learning with Noisy Labels

Kun Yi, Guo-Hua Wang, and Jianxin Wu, *Member, IEEE*

Abstract—Deep learning has achieved excellent performance in various computer vision tasks, but requires a lot of training examples with clean labels. It is easy to collect a dataset with noisy labels, but such noise makes networks overfit seriously and accuracies drop dramatically. To address this problem, we propose an end-to-end framework called PENCIL, which can update both network parameters and label estimations as label distributions. PENCIL is independent of the backbone network structure and does not need an auxiliary clean dataset or prior information about noise, thus it is more general and robust than existing methods and is easy to apply. PENCIL can even be used repeatedly to obtain better performance. PENCIL outperforms previous state-of-the-art methods by large margins on both synthetic and real-world datasets with different noise types and noise rates. And PENCIL is also effective in multi-label classification tasks through adding a simple attention structure on backbone networks. Experiments show that PENCIL is robust on clean datasets, too.

Index Terms—Recognition, Deep Learning, Label Noise, Multi-Label.

1 Introduction

Deep learning has shown very impressive performance on various vision problems, e.g., classification, detection and semantic segmentation. Although there are many factors for the success of deep learning, one of the most important is the availability of large-scale datasets with clean annotations like ImageNet [1].

However, collecting a large scale dataset with clean labels is expensive and time-consuming. On one hand, expert knowledge is necessary for some datasets such as the fine-grained CUB-200 [2], which demands knowledge from ornithologists. On the other hand, we can easily collect a large scale dataset with noisy annotations from various websites [3], [4], [5]. These noisy annotations can be obtained by extracting labels from the surrounding texts or using the searching keywords [6]. For a huge dataset like JFT300M (which contains 300 million images), it is impossible to manually label it and inevitably about 20% noisy labels exist in this dataset [7]. Hence, being able to deal with noisy labels is essential.

The label noise problem has been studied for a long time [8], [9]. Along with the recent successes of various deep learning methods, noise handling in deep learning has gained momentum, too [6], [10], [11]. However, existing methods often have prerequisites that may not be practical in many applications, e.g., an auxiliary set with clean labels [6] or prior information about the noise [12]. Some methods are very complex [13], which hurts their deployment capability. Overfitting to noise is another serious difficulty. For a DNN with enough capacity, it can memorize the random labels [14]. Thus, some noise handling methods may finally still overfit and their performance decline seriously, i.e., they are not robust. Their accuracies on the clean test set reach a peak in the middle of the training process, but will degrade afterwards and the accuracies after the final training epoch are poor [12], [15].

We attack the label noise problem from two aspects. First, we model the label for an image as a distribution among all possible labels [16] instead of a fixed categorical value. This *probabilistic* modeling lends us the flexibility to handle noise-contaminated and noise-free labels in a *unified* manner. Second, inspired by [17], we maintain and update the label distributions in both network parameter learning (in which label distributions act as labels) and label learning (in which label distributions are updated to correct noise). Unlike [17] which updates labels simply by using the running average of network predictions, we correct noise and update our label distributions in a principled *end-to-end* manner. The proposed framework is called PENCIL, meaning *probabilistic end-to-end noise correction in labels*. The PENCIL framework only uses the noisy labels to initialize our label distributions, then iteratively correct the noisy labels by updating the label distributions, and the network loss function is computed using the label distributions rather than the noisy labels.

Our contributions are as follows.

- We propose an end-to-end framework PENCIL for noisy label handling. PENCIL is independent of the backbone network structure and does not need an auxiliary clean dataset or prior information about noise, thus it is easy to apply. PENCIL utilizes back-propagation to probabilistically update and correct image labels in addition to updating the network parameters. To the best of our knowledge, PENCIL is the first method in this line.
- We propose a variant of the DLDL method [16], which is essential for correcting noise contained in our label distributions. PENCIL achieves state-of-the-art accuracy on datasets with both synthetic and real-world noisy labels (e.g., CIFAR-10, CIFAR-100 and Clothing1M). We also propose an attention structure and extend the PENCIL framework to handle multi-label tasks without *or* with label noise.

- *This research was partially supported by the National Natural Science Foundation of China (61772256, 61422203).*

- *K. Yi, G.-H. Wang and J. Wu are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: {yik,wangguohua,wujx}@lamda.nju.edu.cn. J. Wu is the corresponding author.*

Manuscript received

- Unlike DLDL, we use inverse KL-divergence in our method. And we show that inverse KL-divergence is indeed more suitable for noise correction than the original KL-divergence.
- PENCIL is robust. It is not only robust in learning with noisy labels, but also robust enough to apply in datasets with zero or small amount of *potential* label noise (e.g., CUB-200) to improve accuracy.

A preliminary version of the PENCIL framework has appeared as a conference publication [18].

2 Related Works

We first briefly introduce related works that inspired this work and other noise handling methods in the literature.

Deep label distribution learning (DLDL) was introduced in [16], which was proposed to handle label *uncertainty* by converting a categorical label (e.g., 25 years old) into a label distribution (e.g., a normal distribution whose mean is 25 and standard deviation is 3). The DLDL method uses constant label distributions and the Kullback-Leibler divergence to compute the network loss. In PENCIL, we use label distributions for a different purpose such that the label distributions can be updated and hence noise can be probabilistically corrected. The original DLDL method did not work in our setup and we designed a new loss function in PENCIL to overcome this difficulty.

For deep learning methods, [14] showed that a deep network with large enough capacity can memorize the training set labels even when they are randomly generated. Hence, they are particularly susceptible to noisy labels. Label noise can lead to serious overfitting and dramatically reduce network accuracy. However, [17] observed that when the learning rate is high, DNNs may maintain relatively high accuracy (i.e., the impact of label noise is not significant). This observation was utilized in [17] to maintain an estimate of the labels using the running average of network predictions with a large learning rate. Then, these estimates were used as supervision signals to train the network. PENCIL is inspired by this observation and [17], too.

Label noise is an important issue and has long been researched [8], [9]. There are mainly two types of label noise: symmetric noise and asymmetric noise, which are modeled in [19] and [11], respectively. [20] is a survey of relatively early methods. [21] argued that deep neural networks are inherently robust to label noise to some extent. And, deep methods have achieved state-of-the-art results in recent years. Hence, we mainly focus on noise handling in deep learning models in this section.

One intuitive and easy solution is to delete all the samples which are considered as unreliable [22]. However, many difficult samples will be deleted, but these samples are important to an algorithm's accuracy [23]. Thus, more profound noisy label handling methods become necessary.

There are mainly two lines of attack to the the noisy label problem: constructing a special model based on noisy labels or using a robust loss function. The objective of these methods is to construct a noise-aware model which explicitly deals with noisy labels. [6] constructed a model to deal with noisy labels, and tested their method on a real-world dataset collected by them. [15] proposed a framework called CNN-CRF, which combined convolutional neural networks (CNN) with conditional random fields (CRF) to characterize noisy labels. [13] utilized similar ideas to determine the confidence of each label. This approach is gaining popularity in recent years (e.g., in [24], [25], [26]), and different techniques such as local inherent dimensionality have been brought into the noisy label learning domain.

Another effective approach is to design robust loss functions in order for a noise-tolerant model. Forward and backward methods [12] explicitly modeled the noise transition matrix in loss computation. [27] investigated the robustness of different loss functions, such as the mean squared loss, mean absolute loss and cross entropy loss. [28] combined advantages of the mean absolute loss and cross entropy loss to obtain a better loss function.

[17] did not fall in these two categories. It is special in the sense that it replaced the noisy label with their own estimate of the label (i.e., running average of the network's predictions). This approach is effective in noise handling but ad-hoc. PENCIL is partly inspired by this work, but more principled and effective.

Existing methods usually have prerequisites that are impractical, such as demanding an additional clean dataset (e.g., to curb overfitting) or a ground-truth noise transition matrix. When these prerequisites are not satisfied, they often fail to produce robust models. These methods are sometimes too complex to be deployed in real-world applications. In contrast, the proposed PENCIL method does not require additional information, and it can be easily applied to any backbone network.

3 The Proposed PENCIL Method

First of all, we define the notations for our study. Column vectors are denoted in bold (e.g., $\boldsymbol{x}$) and matrices in capital form (e.g., X). Specifically, $\mathbf{1}$ is a vector of all-ones. We use both hard labels and soft labels. The hard-label space is $\mathcal{H} = \{\boldsymbol{y} : \boldsymbol{y} \in \{0,1\}^c, \mathbf{1}^\top \boldsymbol{y} = 1\}$, and the soft-label space is $\mathcal{S} = \{\boldsymbol{y} : \boldsymbol{y} \in [0,1]^c, \mathbf{1}^\top \boldsymbol{y} = 1\}$. That is, a soft-label is a label distribution.

3.1 Probabilistic Modeling of Noisy Labels

In a c-class classification problem, we have a training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$. In the ideal scenario, every image $\boldsymbol{x}_i$ has a clean label $\boldsymbol{y}_i \in \mathcal{H}$, which is a one-hot vector (i.e., equivalent to an integer between 1 and c). In our noisy label problem, the labels might be wrong with relatively high probability and we use $\hat{\boldsymbol{y}}_i \in \mathcal{H}$ to denote labels which may contain noise. Using cross entropy, the loss function is

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} \hat{y}_{ij} \log f_j(\boldsymbol{x}_i; \boldsymbol{\theta}) \,, \tag{1}$$

where $\hat{y}_{ij}$ is the j'th element of $\hat{\boldsymbol{y}}_i$, f is a model's prediction (processed by the softmax function) and $\boldsymbol{\theta}$ is the set of network parameters.

In PENCIL, we maintain a label distribution $\boldsymbol{y}_i^d \in \mathcal{S} = \{\boldsymbol{y} : \boldsymbol{y} \in [0,1]^c, \mathbf{1}^\top \boldsymbol{y} = 1\}$ for every image $\boldsymbol{x}_i$, which is our estimate of the *underlying noise-free* label for $\boldsymbol{x}_i$. $\boldsymbol{y}_i^d$ is used as the pseudo-ground-truth label in our learning, which is

initialized based on the noisy label $\hat{\boldsymbol{y}}_i$. It is continuously updated (i.e., the noise is gradually corrected) *through back-propagation*. This probabilistic setting allows ample flexibility for noise correction. Note that our probabilistic modeling of the noisy labels is different from that in DLDL [16]. Label distributions in DLDL are fixed and cannot be updated.

In [16], the loss function is KL-divergence:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} KL(\boldsymbol{y}_i^d || f(\boldsymbol{x}_i;\boldsymbol{\theta})), \text{and} \tag{2}$$

$$\mathrm{KL}(\boldsymbol{y}_i^d || f(\boldsymbol{x}_i;\boldsymbol{\theta})) = \sum_{j=1}^{c} y_{ij}^d \log\left(\frac{y_{ij}^d}{f_j(\boldsymbol{x}_i;\boldsymbol{\theta})}\right). \tag{3}$$

This loss is used in [17], too. However, KL-divergence is an asymmetric function. Hence, if we exchange the two operands in Eq. 2, we obtain a new loss function

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} KL(f(\boldsymbol{x}_i;\boldsymbol{\theta}) || \boldsymbol{y}_i^d), \text{and} \tag{4}$$

$$\mathrm{KL}(f(\boldsymbol{x}_i;\boldsymbol{\theta}) || \boldsymbol{y}_i^d) = \sum_{j=1}^{c} f_j(\boldsymbol{x}_i;\boldsymbol{\theta}) \log\left(\frac{f_j(\boldsymbol{x}_i;\boldsymbol{\theta})}{y_{ij}^d}\right). \tag{5}$$

We will soon show that Eq. 4 is more suitable for noise handling. In fact, Eq. 2 led to very poor results in our experiments and we propose to use Eq. 4 as one of the loss functions in PENCIL. More details will be discussed in Section 3.4.

3.2 End-to-end Noise Correction in Labels

Our label distribution $\boldsymbol{y}^d$ models the unknown noise-free label for $\boldsymbol{x}_i$. Hence, we need to estimate these distributions in our learning process. Let $\boldsymbol{X}$ and $\boldsymbol{Y}^d$ be the union of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i^d$ (for all $1 \le i \le n$), respectively. Different from [17], we let $\boldsymbol{Y}^d$ be part of the parameters that are to be updated in the back-propagation process. That is, PENCIL not only updates the network parameters $\boldsymbol{\theta}$ as in traditional networks, but also updates $\boldsymbol{Y}^d$ (i.e., $\boldsymbol{y}_i^d$) in every iteration. Therefore, we optimize both network parameters and label distributions as follows:

$$\min_{\boldsymbol{\theta},\boldsymbol{Y}^d} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{Y}^d | \boldsymbol{X}) \tag{6}$$

The overall architecture of PENCIL is shown in Fig. 1.

In the PENCIL framework, three types of "labels" ($\boldsymbol{y}^d$, $\hat{\boldsymbol{y}}$ and $\tilde{\boldsymbol{y}}$) are involved. Label distribution $\boldsymbol{y}^d$ is updated by back-propagation. In the end, $\boldsymbol{y}^d$ will be a good estimate of the underlying unknown noise-free label (i.e., noise corrected label). $\tilde{\boldsymbol{y}}$ is a variable that assists $\boldsymbol{y}^d$ to be normalized to a probability distribution, by

$$\boldsymbol{y}^d = \mathrm{softmax}(\tilde{\boldsymbol{y}}). \tag{7}$$

Hence, $\tilde{\boldsymbol{y}}$ is not constrained and can be updated freely using back-propagation, but $\boldsymbol{y}^d$ is always a valid distribution.

The original noisy label $\hat{\boldsymbol{y}}$ does not directly impact the parameter ($\boldsymbol{\theta}$) learning. However, it is useful because we use it to indirectly initialize our label distribution $\boldsymbol{y}^d$. At the start of PENCIL, $\tilde{\boldsymbol{y}}$ is initialized by $\hat{\boldsymbol{y}}$ as follows:

$$\tilde{\boldsymbol{y}} = K\hat{\boldsymbol{y}}, \tag{8}$$

where K is a large constant ($K = 10$ in our experiments), and hence from Eq. 7 we have $\boldsymbol{y}^d \approx \hat{\boldsymbol{y}}$ after this initialization.

3.3 Compatibility Loss

The noisy label $\hat{\boldsymbol{y}}$ is also useful in PENCIL's loss computation. In fact, there are lots of (e.g., 80% of) correct labels even in datasets with noisy labels. Therefore, we should not let the estimated label distribution $\boldsymbol{y}^d$ be completely different from those noisy labels $\hat{\boldsymbol{y}}$.

We define a compatibility loss $\mathcal{L}_o(\hat{\boldsymbol{Y}}, \boldsymbol{Y}^d)$ to enforce this requirement, as

$$\mathcal{L}_o(\hat{\boldsymbol{Y}}, \boldsymbol{Y}^d) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} \hat{y}_{ij} \log y_{ij}^d, \tag{9}$$

which is a classic cross entropy loss between label distribution and noisy label.

3.4 Classification Loss

Selection of the classification loss is very important, because we will update the value of $\boldsymbol{y}^d$ by its gradient directly. And according to Eq. 7, we have:

$$\frac{\partial y_{ik}^d}{\partial \tilde{y}_{ij}} = y_{ik}^d(\delta_{k=j} - y_{ij}^d), \tag{10}$$

in which $\delta_{k=j} = 1$ if $k = j$, and equals 0 if $k \neq j$.

Assume that $\mathcal{L}_c$ is an *arbitrary* classification loss function, and that we only consider one single input, we then can calculate the gradient of $\tilde{\boldsymbol{y}}$ as:

$$\frac{\partial \mathcal{L}_c}{\partial \tilde{y}_j} = \sum_{k=1}^{c} \frac{\partial \mathcal{L}_c}{\partial y_k^d}\frac{\partial y_k^d}{\partial \tilde{y}_j} \tag{11}$$

$$= \sum_{k=1}^{c} \frac{\partial \mathcal{L}_c}{\partial y_k^d} y_k^d(\delta_{k=j} - y_j^d) \tag{12}$$

$$= \sum_{k=1}^{c} \delta_{k=j} y_k^d \frac{\partial \mathcal{L}_c}{\partial y_k^d} - y_{ij}^d \sum_{k=1}^{c} y_k^d \frac{\partial \mathcal{L}_c}{\partial y_k^d} \tag{13}$$

$$= y_j^d \frac{\partial \mathcal{L}_c}{\partial y_j^d} - y_j^d \sum_{k=1}^{c} y_k^d \frac{\partial \mathcal{L}_c}{\partial y_k^d}. \tag{14}$$

Denote $g_j = y_j^d \frac{\partial \mathcal{L}_c}{\partial y_j^d}$, then we have:

$$\sum_{k=1}^{c} y_k^d \frac{\partial \mathcal{L}_c}{\partial y_k^d} = \sum_{k=1}^{c} g_k = \boldsymbol{g}^T\mathbf{1}, \tag{15}$$

$$\frac{\partial \mathcal{L}_c}{\partial \tilde{y}_j} = g_j - \boldsymbol{g}^T\mathbf{1} y_j^d, \tag{16}$$

$$\sum_{j=1}^{c} \frac{\partial \mathcal{L}_c}{\partial \tilde{y}_j} = \sum_{j=1}^{c} g_j - \boldsymbol{g}^T\mathbf{1}\sum_{j=1}^{c} y_j^d \tag{17}$$

$$= \boldsymbol{g}^T\mathbf{1} - \boldsymbol{g}^T\mathbf{1}(\boldsymbol{y}^d)^T\mathbf{1}. \tag{18}$$

We know $(\boldsymbol{y}^d)^T\mathbf{1} = 1$ because $\boldsymbol{y}^d$ is a label distribution, thus $\sum_{j=1}^{c} \frac{\partial \mathcal{L}_c}{\partial \tilde{y}_j} = 0$. Then, we get the following proposition:

Proposition 1. *The sum of all dimensions in the gradient of $\mathcal{L}_c$ with respect to $\boldsymbol{y}_d$ is zero.*

In Section 3.1 we mentioned the difference between KL- and inverse KL-divergence, now we discuss their suitability as the classification loss function in PENCIL based on the above derivation.

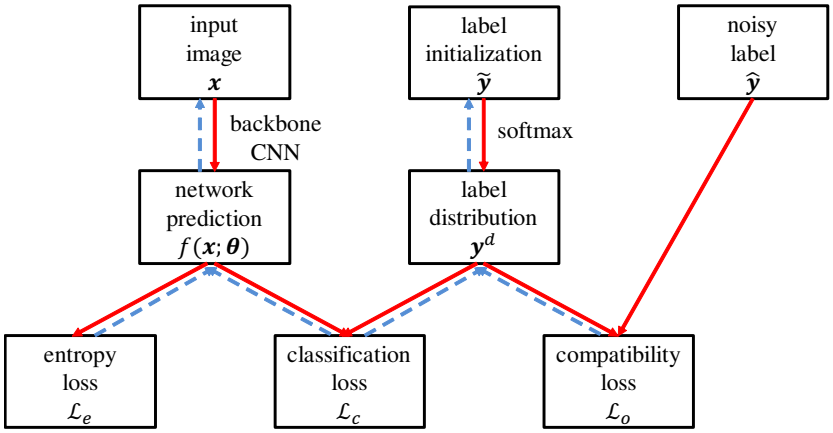

Figure 1. The PENCIL learning framework. We use label distributions $\boldsymbol{y}^d$ (which is the softmax transformed version of label initialization variables $\tilde{\boldsymbol{y}}$) to replace noisy labels $\hat{\boldsymbol{y}}$. The label distributions are updated in every iteration using three loss functions, among which the classification loss and compatibility loss updates $\boldsymbol{y}^d$ by requiring the label distributions produce both smooth models and not too distant from the noisy labels.

3.4.1 Case 1

When the classification loss is the KL-divergence, we have:

$$\frac{\partial \mathcal{L}_c}{\partial y_j^d} = 1 + \log \frac{y_j^d}{f_j(\boldsymbol{x};\boldsymbol{\theta})} \,. \tag{19}$$

Then,

$$g_j = y_j^d + y_j^d \log \frac{y_j^d}{f_j(\boldsymbol{x};\boldsymbol{\theta})} \,, \tag{20}$$

$$\boldsymbol{g}^T \mathbf{1} = \sum_{j=1}^{c} y_j^d + \sum_{j=1}^{c} y_j^d \log \frac{y_j^d}{f_j(\boldsymbol{x};\boldsymbol{\theta})} \tag{21}$$

$$= 1 + \mathcal{L}_c \,. \tag{22}$$

Substituting Eq. 20 and Eq. 22 into Eq. 16 we get the following result:

$$\frac{\partial \mathcal{L}_c}{\partial \tilde{y}_j} = y_j^d + y_j^d \log \frac{y_j^d}{f_j(\boldsymbol{x};\boldsymbol{\theta})} - (1 + \mathcal{L}_c) y_j^d \tag{23}$$

$$= y_j^d \left(\log \frac{y_j^d}{f_j(\boldsymbol{x};\boldsymbol{\theta})} - \mathcal{L}_c \right) \,. \tag{24}$$

3.4.2 Case 2

When the classification loss is the inverse KL-divergence, we have:

$$\frac{\partial \mathcal{L}_c}{\partial y_j^d} = \frac{f_j(\boldsymbol{x};\boldsymbol{\theta})}{y_j^d} \,. \tag{25}$$

Then,

$$g_j = -y_j^d \frac{f_j(\boldsymbol{x};\boldsymbol{\theta})}{y_j^d} = -f_j(\boldsymbol{x};\boldsymbol{\theta}) \,. \tag{26}$$

Substituting Eq. 26 into Eq. 16 we get the following result:

$$\frac{\partial \mathcal{L}_c}{\partial \tilde{y}_j} = -f_j(\boldsymbol{x};\boldsymbol{\theta}) + f_j(\boldsymbol{x};\boldsymbol{\theta})^T \mathbf{1} y_j^d \tag{27}$$

$$= y_j^d - f_j(\boldsymbol{x};\boldsymbol{\theta}) \,. \tag{28}$$

Next we compare Eq. 24 and Eq. 28. We see that if y_j^d is almost zero, the value of Eq. 24 is also almost zero but the value of Eq. 28 is a negative value which depends on the prediction of the network. This difference tells us that the KL-divergence is not suitable for noise correction, but our proposed inverse KL-divergence is.

When we use the original KL-divergence, if the original noisy label is wrong, the value corresponding to the correct label $y_{correct}^d$ is almost zero, then the gradient of it is almost zero, too. Therefore we cannot correct the wrong original label (i.e., cannot successfully increase $y_{correct}^d$). Finally we also cannot get the correct label distribution. So the original KL-divergence is not suitable. But when we use the inverse KL-divergence, we can successfully update the $\boldsymbol{y}^d$ and may get the correct label distribution.

3.5 Entropy Loss

Obviously, when the prediction $f(\boldsymbol{x};\boldsymbol{\theta})$ is the same as the label distribution $\boldsymbol{y}^d$, the network will stop updating. However, $f(\boldsymbol{x};\boldsymbol{\theta})$ tends to approach $\boldsymbol{y}^d$ fairly quickly, because label distributions are used as the supervision signal for learning network parameters $\boldsymbol{\theta}$. Following [17], we add an additional loss (regularization) term to avoid this problem. The entropy loss can force the network to peak at only one category rather than being flat because the one-hot distribution has the smallest possible entropy value. This property is advantageous for classification problems. The entropy loss is defined as

$$\mathcal{L}_e(f(\boldsymbol{x};\boldsymbol{\theta})) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} f_j(\boldsymbol{x};\boldsymbol{\theta}) \log f_j(\boldsymbol{x};\boldsymbol{\theta}) \,. \tag{29}$$

At the same time, it also helps avoid the training from being stalled in our PENCIL framework, because the label distribution is not going to be a one-hot distribution and then $f(\boldsymbol{x};\boldsymbol{\theta})$ will be different from $\boldsymbol{y}^d$.

3.6 The Overall PENCIL Framework

With all components ready, the PENCIL loss function is

$$\mathcal{L} = \frac{1}{c}\mathcal{L}_c(f(\boldsymbol{x};\boldsymbol{\theta}), \boldsymbol{Y}^d) + \alpha \mathcal{L}_o(\hat{\boldsymbol{Y}}, \boldsymbol{Y}^d) + \frac{\beta}{c}\mathcal{L}_e(f(\boldsymbol{x};\boldsymbol{\theta})) \,,$$

in which α and β are two hyperparameters. Using this loss function and the PENCIL framework's architecture in

Algorithm 1 The proposed PENCIL framework

Input: the noisy training set $\{\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i\}$ $(1 \leq i \leq n)$, and the number of training epochs T

1: initialize $\tilde{\boldsymbol{y}}_i$ $(1 \leq i \leq n)$ by Eq. 8
2: $t \leftarrow 1$
3: **while** $t \leq T$ **do**
4: update $\boldsymbol{\theta}$ and $\boldsymbol{y}_i^d$ $(1 \leq i \leq n)$ by forward computation and backward propagation in the mini-batch fashion using all n training examples (i.e., to finish one epoch)
5: $t \leftarrow t + 1$

Output: the trained network model $\boldsymbol{\theta}$, and the noise corrected labels $\boldsymbol{y}_i^d$ $(1 \leq i \leq n)$.

Fig. 1, we can use *any* deep neural network as the backbone network in Fig. 1, and then equip it with the PENCIL component to handle learning problems with noisy labels. The relationship between variables and loss functions are clearly visualized in Fig. 1 as arrows. Forward computations are visualized by red solid arrows, while back-propagation computations are visualized as blue dashed arrows. The algorithmic description of the PENCIL framework is shown in Algorithm 1.

We want to add two notes about PENCIL. First, the error back-propagation process in PENCIL is pretty straightforward. For example, it can be done automatically in deep learning packages that support automatic gradient computation. Second, after the network has been fully trained (cf. Section 4), those PENCIL-related components in Fig. 1 are *not needed* at all—the backbone network alone can perform prediction for future test examples.

Similar to [17], we implement our PENCIL training through 3 steps.

Backbone learning: We firstly train the backbone network with a large fixed learning rate from scratch without noise handling. As aforementioned, it is observed that when the learning rate is high, a DNN often does not overfit the label noise. Therefore, in this step, we use a fixed high learning rate with only the cross-entropy loss function in Eq. 1. The resulted DNN is the backbone network in Fig. 1.

PENCIL learning: Then, we use the PENCIL framework to update both network parameters and label distributions. The learning rate is still a fixed high value. Therefore, the network will not overfit label noise and the label distributions will correct noise in the original labels. At the end of this step, we obtain a label distribution vector for every image. Algorithmic details are shown in Algorithm 1. Note that in practice we find that updating $\tilde{\boldsymbol{y}}$ requires a learning rate that is much larger than that used for updating other parameters. Because the overall learning rate is fixed in this step, we simply use one single hyperparameters λ to update $\tilde{\boldsymbol{y}}$ (i.e., do not use PENCIL's overall learning rate), as

$$\tilde{\boldsymbol{y}} \leftarrow \tilde{\boldsymbol{y}} - \lambda \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{y}}}. \tag{30}$$

Final fine-tuning: Lastly, we use the learned label distributions to fine-tune the network using only the classification loss $\mathcal{L}_c$ (i.e., $\alpha = \beta = 0$). In this step, the label distributions will not be updated and the learning rate will be gradually reduced as in common neural network training.

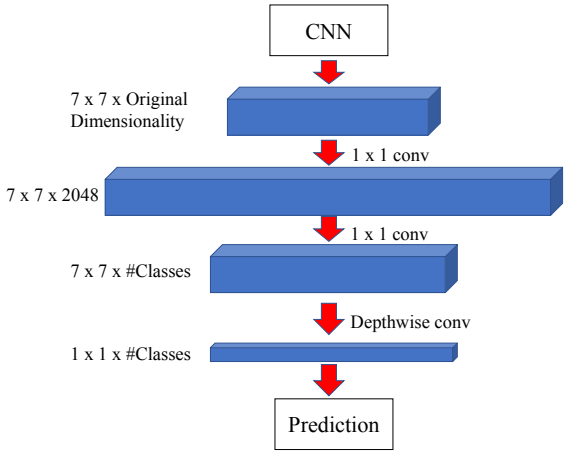

Figure 2. The proposed attention structure for multi-label classification. This component can replace the global average pooling layer in arbitrary backbone network.

3.7 Attention Structure for Multi-label Classification

When we extend the problem from single-label to multi-label, the complexity of the problem increased a lot. Therefore, the original network structure together with sigmoid loss function is too simple to handle this situation. Our PENCIL framework corrects noisy labels based on the predictions provided by the backbone network. Therefore, a better backbone network is very important.

We propose a simple attention structure to replace the global average pooling layer in the original backbone network. The overall framework of this attention structure is shown in Fig. 2.

This structure firstly increases the feature dimensionality (e.g., from 512 to 2048) by a 1×1 conv layer. This high dimensional feature can encode more information. Then, we change the feature dimensionality to the number of classes via another 1×1 conv layer. For this output (i.e., a $7 \times 7 \times \#\text{Classes}$ activation map), we regard one dimension as corresponding to one category. With this one-to-one correspondence, each category has a unique feature matrix (i.e., a 7×7 matrix), with each of the 49 values corresponding to predictions based on different sub-regions in the original input image. We can incorporate these predictions from different sub-regions by combining these 49 values to compute one probability for the corresponding category. Therefore we use a depthwise conv layer to compute one value corresponding to one category individually for each dimension. Then we use the sigmoid function to normalize these values. This final result is our final prediction.

In this proposed component, each feature matrix only *pays attention to* one specific category, hence we call this component a (spatial) *attention structure*.

3.8 Repetitive Training

The proposed PENCIL framework corrects noisy labels based on the predictions of the last epoch. The initial backbone model has great impact to the PENCIL learning process. However, the initial backbone network is only trained with a fixed learning rate. The model obtained by our PENCIL

framework is much better than the model obtained by the first step of the PENCIL framework. So we can use it to replace the model obtained by backbone learning. Next we process the PENCIL learning and the final fine-tuning again [29]. The above process is called repetitive training. We can even repeat this process multiple times to achieve better performance.

After using repetitive training, the whole PENCIL framework includes one time of backbone learning, and multiple times of PENCIL learning and final fine-tuning. It is worth noting that different from [29], we just used the final model as the backbone model but do not use the final label distributions to initialize the $\boldsymbol{y}^d$. Because if some labels are still wrong in the final label distributions, the errors may accumulate through the repetitive training. We want to avoid the rapid propagation of impacts of these wrong labels.

4 Experiments

We tested the proposed PENCIL framework on both synthetic and real-world datasets: CIFAR-100 [30], CIFAR-10 [30], CUB-200 [2] and Clothing1M [6]. And we also tested it on multi-label datasets without *or* with noisy label: MS-COCO [31] and Open Images [32]. All experiments were implemented using the PyTorch framework.

4.1 Datasets

CIFAR-100: Following [28], we retained 10% of the training data as the validation set, and *both* train and validation sets were noise contaminated. However, we did *not* use the validation set in our method, because PENCIL *does not need a validation set*.

There are two types of noises: symmetric and asymmetric. Following [28], in the symmetric noise setup, label noise is uniformly distributed among all categories, and the label noise percentage is $r \in [0, 1]$. For every example, if the correct label is i, then the noise-contaminated label has $1 - r$ probability to remain correct, but has r probability to be drawn uniformly from the c labels. The asymmetric noise label was generated by flipping each class to the next class circularly with noise rate $r \in [0, 1]$.

CIFAR-10: Following [17], we retained 10% of the CIFAR-10 training data as the validation set and modify the original correct labels to obtain different noisy label datasets. The setting for symmetric noise is the same as that in CIFAR-100. As for asymmetric noise, following [12] the noisy labels were generated by mapping `truck` $\rightarrow$ `automobile`, `bird`$\rightarrow$ `airplane`, `deer` $\rightarrow$ `horse` and `cat` $\leftrightarrow$ `dog` with probability r. These noise generation methods are in coincidence with confusions that often happen in the real world.

Clothing1M: Clothing1M is a large-scale dataset with noisy labels. It consists of more than one million images from 14 classes with many wrong labels. Images were obtained from several online shopping websites and labels were generated by their surrounding texts. The estimated noise level is roughly 40% [6]. This dataset is seriously imbalanced and the label mistakes mostly happen between similar classes (i.e., asymmetric). There exist additional training, validation and test sets with 50k, 14k and 10k examples whose labels are believed to be clean, respectively.

CUB-200: We tested the robustness of our framework in a fine-grained classification dataset CUB-200. CUB-200 contains 11788 images of 200 species of birds, which is not considered to have the noisy label difficulty. Therefore, we tested our framework on this dataset to show that PENCIL is robust. In addition, there is probably a small percentage of noisy labels in CUB-200 [33]. It is interesting to observe whether PENCIL is robust and effective in such a dataset.

MS-COCO: MS-COCO is originally collected for object detection tasks, and it's also widely used for the multi-label classification task. MS-COCO consists of 122218 images from 80 classes which are common in real world. It contains two subsets: a training set with 82081 images and a validation set with 40137 images. We will use it to show the effectiveness of the proposed attention structure.

Open Images: Open Images is a dataset collected from real world. It consists of about 9 million images annotated with image-level labels, object bounding boxes, object segmentation masks, and visual relationships. It can be used as a multi-label classification dataset with noisy labels. Each image has two kinds of labels: human-verified and machine-generated. It can be considered that the human-verified labels are almost correct and the machine-generated labels contain various levels of noise. But the number of the former is much less than that of the latter (27.9M human-verified vs. 78.9M machine-generated labels). Thus, if we use this dataset, we have to make use of these machine-generated labels to ensure enough training data. We randomly selected a subset of 250 categories from this dataset for our experiments. On this dataset, we show the effectiveness of our proposed PENCIL framework and attention structure.

4.2 Implementation Details

Next, we describe more implementation details for each dataset.

CIFAR-100: We used ResNet-34 [34] as the backbone network for fair comparison with existing methods. The learning rate was 0.35, $\alpha = 0.1$, $\beta = 0.4$, and $\lambda = 10000$. Mean subtraction, horizontal random flip and 32×32 random crops after padding 4 pixels on each side were performed as data preprocessing and augmentation. We used SGD with 0.9 momentum, a weight decay of 10^{-4}, and batch size of 128. Following [17], the epoch numbers for three steps were 70, 130 and 120, respectively. In the last step, we used the learning rate of 0.2 and divided it by 10 after 40 and 80 epochs [17]. All experiments on CIFAR-100 used the same settings as described above. In fact, we can obtain better results by further tuning the hyperparameters (e.g., as what we will soon introduce for CIFAR-10). However, we choose to use the same set of hyperparameters to demonstrate the robustness of our framework.

CIFAR-10: We used PreAct ResNet-32 [35] as the backbone network for fair comparison with existing methods. We used the same settings as those for CIFAR-100, except the overall learning rate, α, β and λ hyperparameters. On CIFAR-10, these hyperparameters are shown in Table 1.

As shown in Table 1, the learning rate increases as the noise rate increases for symmetric noise. This is reasonable, because when noise rate gets higher, we need stronger robustness and we can increase the learning rate to prevent

Table 1
Hyperparameters for CIFAR-10 experiments. 3000 $\rightarrow$ 0 means that λ decreases from 3000 to 0 linearly.

Symmetric Noise				
noise rate (%)	learning rate	α	β	λ
10	0.02	0.1	0.8	200
30	0.03	0.1	0.8	300
50	0.04	0.1	0.8	400
70	0.08	0.1	0.8	800
90	0.12	0.1	0.4	1200
Asymmetric Noise				
noise rate (%)	learning rate	α	β	λ
10	0.06	0.1	0.4	600
20	0.06	0.1	0.4	600
30	0.06	0.1	0.4	600
40	0.03	0	0.4	3000 $\rightarrow$ 0
50	0.03	0	0.4	4000 $\rightarrow$ 0

our network from overfitting. And, when the noise rate is very high (e.g., 50% asymmetric), there are too many noisy labels. Hence, we can remove the effect of noisy labels by removing $\mathcal{L}_o$ (i.e., set α to 0). At the same time, we require a large λ to correct these noisy labels quickly. However, after a few epochs, the noisy labels were quickly corrected to a stable state (cf. Fig. 2 and Fig. 3). Hence, we need to decrease λ linearly to prevent wrong updates in later epochs.

CUB-200: On this dataset, we used ResNet-50 [34] pre-trained on ImageNet. Data preprocessing and augmentation is also applied, including performing mean subtraction, horizontal random flip, resizing the image to 256×256 and 224×224 random crops. We used SGD with 0.9 momentum, a weight decay of 10^{-4}, and batch size of 16. The number of epochs for the three steps are 35, 65 and 60, respectively. The learning rate of the first and second step is 2×10^{-3}. In the last step, the learning rate is 10^{-3} and divided by 10 after 20 epochs and 40 epochs. β is 0.8 and we reported results for different values of α and λ as ablation studies.

Clothing1M: We used ResNet-50 pre-trained on ImageNet as the backbone network for fair comparison with existing methods. Data preprocessing and augmentation are the same as those in CUB-200. We used SGD with 0.9 momentum, a weight decay of 10^{-3}, and batch size of 32. The epoch numbers of the three steps are 5, 10 and 10, respectively. The first step learning rate is 1.6×10^{-3} and the second step learning rate is 8×10^{-4}. The last step learning rate is 5×10^{-4} and divided by 10 after 5 epochs. $\alpha = 0.08$, $\beta = 0.8$. In first 5 epochs of second step $\lambda = 3000$, and in last 5 epochs of second step $\lambda = 500$.

This dataset exists serious data imbalance. Therefore, we randomly selected a small balanced subset (using the noisy labels) to relieve the difficulty caused by imbalance. The small subset includes about 260k images and all classes have the same number of images. All our experiments on Clothing1M were done with this subset in this study. However, note that this subset is not truly balanced, because the labels are noisy.

MS-COCO: On this dataset, we used efficient-b0 to -b5 [36] as our backbone networks. We compared the performance of backbone networks with and without our attention structure on this dataset. The input size of different networks followed their official setting. For convenience, data preprocessing and augmentation just include mean subtraction and horizontal random flip. We used SGD with 0.9 momentum, a weight decay of 10^{-4}, and batch size of 24, the number of epochs is 60. The initial learning rate is 0.01 and divided by 10 every 20 epochs. In addition, the loss function is changed from cross entropy to binary cross entropy for multi-label classification.

Open Images: We randomly selected a subset of 250 categories from Open Images. Every category has about 500 training images, 100 validation images and 50 test images, respectively. On this dataset, the number of human-verified labels is much smaller than that of machine-generated ones. So we used the machine-generated labels which contain various levels of noise to ensure enough images in our training set. But our validation and test set only used the images with human-verified labels.

We used efficientnet-b0 as our backbone network. Data preprocessing and augmentation are the same as those in CUB-200. We used RMSprop following the common practice on this dataset, with 0.9 momentum, 0.9 alpha, a weight decay of 4×10^{-5}, and the batch size of 128. In the baseline, the number of epochs is 90. The learning rate is 0.1 and multiplied by 0.94 every 2 epochs. In baseline with attention structure, the number of epochs is also 90. The basic learning rate is 0.05 and also multiplied by 0.94 every 2 epochs. In addition, the learning rate of our attention structure is 5 times the basic learning rate. In PENCIL framework with attention structure, $\alpha = 0.2$, $\beta = 0.2$ and $\lambda = 300$. The epoch numbers of the three steps are 35, 65 and 60, respectively. The learning rate of first and second steps is 5×10^{-3}. In the last step, the learning rate is also 5×10^{-3} and multiplied by 0.94 every 2 epochs. On this dataset, the machine-generated labels are probability values (i.e., $0, 0.1, 0.2, ..., 1$). We used them and the human-verified labels as our noisy ground-truth labels directly. All the loss functions are changed to binary version (e.g., inverse binary KL-divergence). In addition, we used both inverse binary KL-divergence and binary KL-divergence in our PENCIL framework.

4.3 Experiments on CIFAR-100

Firstly we tested PENCIL on CIFAR-100. The results are shown in Table 2. All dataset settings followed [28]. The method "Forward T [12]" used the ground-truth noise transition matrix (which is not available in real-world datasets), hence its numbers were not compared with other methods. Except for the 80% symmetric noise case, PENCIL significantly outperformed previous methods in all symmetric and asymmetric noise cases. Even if "Forward T" used strong prior information which should not have been used, our PENCIL method still outperformed it in most cases.

As for the 80% symmetric noise case, it revealed a *failure mode* of the proposed PENCIL method. When the noise rate is too high (e.g., 80%), the correct labels only form a minority group and they are too weak to bootstrap the noise correction process. Hence, PENCIL tends to fail in such high noise rate problems. Fortunately, we hardly deal with such high noise rate in real-world applications. For example, the large scale real-world image dataset JFT300M [7] only includes about 20% noisy labels.

We have intentionally chosen the same set of hyperparameters in all experiments on this dataset, and the results demonstrate the *robustness* of our PENCIL framework to

Table 2
Results on CIFAR-100. We reported the average accuracy and standard deviation of 5 trials. $\#1$ to $\#5$ are quoted from [28]. PENCIL ($\#6$) is the result of last epoch (without using the validation set). The row with a star * ($\#2$) did not participate in comparison for fairness.

#	method	Symmetric Noise				Asymmetric Noise			
	noise rate (%)	20	40	60	80	10	20	30	40
1	Cross Entropy Loss	58.72±0.26	48.20±0.65	37.41±0.94	18.10±0.82	66.54±0.42	59.20±0.18	51.40±0.16	42.74±0.61
2	Forward T * [12]	63.16±0.37	54.65±0.88	44.62±0.82	24.83±0.71	71.05±0.30	71.08±0.22	70.76±0.26	70.82±0.45
3	Forward $\hat{T}$ [12]	39.19±2.61	31.05±1.44	19.12±1.95	8.99±0.58	45.96±1.21	42.46±2.16	38.13±2.97	34.44±1.93
4	$\mathcal{L}_q$ [28]	66.81±0.42	61.77±0.24	53.16±0.78	29.16±0.74	68.36±0.42	66.59±0.22	61.45±0.26	47.22±1.15
5	Trunc $\mathcal{L}_q$ [28]	67.61±0.18	62.64±0.33	54.04±0.56	**29.60±0.51**	68.86±0.14	66.59±0.23	61.87±0.39	47.66±0.69
6	PENCIL ($last$)	**73.86±0.34**	**69.12±0.62**	**57.79±3.86**	fail	**75.93±0.20**	**74.70±0.56**	**72.52±0.38**	**63.61±0.23**

these hyperparameters. We can obtain better accuracy by using different hyperparameters for different noise rate and noise type, as shown in Table 1 on the CIFAR-10 dataset.

4.4 Experiments on CIFAR-10

Next, we evaluated the performance of our PENCIL framework on CIFAR-10. All the settings have been described in Section 4.2. On the original noise-free CIFAR-10 dataset, the result of our backbone network (PreAct ResNet-32) is 94.05%. Our setup followed that in [17]. However, results in [17] used a prior knowledge (i.e., all categories have the same number of noise-free training examples), which should not be used. For fair comparison, we implemented the "Tanaka *et al.* [17]" method and in our implementation we did not use this prior knowledge.

Table 3 lists results of symmetric noise for CIFAR-10. In Table 3, "$best$" denotes the test accuracy of the epoch where the validation accuracy was optimal and "$last$" denotes the test accuracy of the last epoch. As aforementioned, when the learning rate is small, the deep neural network's accuracy will decline because the network memorizes all the (noisy) labels, i.e., the network is overfitting. As shown in row $\#1$, the traditional neural network using the classic cross entropy loss is heavily affected by this difficulty. Its $best$-epoch test accuracy was significantly better than that of the $last$-epoch one. And, as the noise rate increased, the gap was even larger because the overfitting to noise became more serious as expected. On the contrary, our method and the Tanaka *et al.* [17] did not have obvious accuracy drop between $best$- and $last$-epochs. Therefore, the proposed PENCIL method has strong robustness. As for the test set accuracy, PENCIL had a clear advantage than competing methods in Table 3. The winning gap became especially apparent when the noise rate increased to larger values. For example, when the noise rate was 90%, PENCIL obtained roughly 7% higher accuracy than that of Tanaka *et al.* and 10% higher than that of cross entropy.

Table 4 lists results of asymmetric noise for CIFAR-10. In terms of robustness, methods shown in row $\#1$, $\#2$ and $\#3$ had the overfitting problem and their test accuracies had large gaps between the $best$- and $last$-epochs. The Tanaka *et al.* method experienced the same issue when the noise rate was high (50%), but was robust in other cases. Our PENCIL method, however, remained robust throughout all the experiments.

The Forward [12] and CNN-CRF [15] methods both require the ground-truth noise transition matrix, which is hardly available in applications. Our method does not require any prior information about noise labels. Table 4 shows that PENCIL has been robust and is the overall accuracy winner on CIFAR-10.

Table 3
Test accuracy on CIFAR-10 with symmetric noise. We reported the average result of 5 trials. All results in this table were based on our own implementation.

#	method	Symmetric Noise					
	noise rate (%)		10	30	50	70	90
1	Cross Entropy Loss	$best$	91.66	89.00	85.15	78.09	50.74
		$last$	88.43	72.78	53.11	33.32	16.30
2	Tanaka *et al.* [17]	$best$	93.23	91.23	88.50	84.51	54.36
		$last$	93.23	91.22	88.51	84.59	53.49
3	PENCIL	$best$	93.26	92.09	90.29	87.10	**61.21**
		$last$	**93.28**	**92.24**	**90.36**	**87.18**	60.80

Table 4
Test accuracy on CIFAR-10 with asymmetric noise. We reported the average result of 5 trials. Rows $\#1$, $\#4$ and $\#5$ were based on our own implementation. Rows $\#2$ and $\#3$ were quoted from [17]. The methods marked with a "*" used additional information that should not be used, and need to be excluded in a fair comparison.

#	method	Asymmetric Noise					
	noise rate (%)		10	20	30	40	50
1	Cross Entropy Loss	$best$	91.09	89.94	88.78	87.78	77.79
		$last$	85.24	80.74	76.09	76.12	71.05
2	Forward T * [12]	$best$	92.4	91.4	91.0	90.3	83.8
		$last$	91.7	89.7	88.0	86.4	80.9
3	CNN-CRF * [15]	$best$	92.0	91.5	90.7	89.5	84.0
		$last$	90.3	86.6	83.6	79.7	76.4
4	Tanaka *et al.* [17]	$best$	92.53	91.89	91.10	91.48	75.81
		$last$	92.64	91.92	91.18	**91.55**	68.35
5	PENCIL	$best$	93.00	**92.43**	**91.84**	91.01	**80.51**
		$last$	**93.04**	**92.43**	91.80	91.16	80.06

We recorded the number of correct labels in PENCIL's second step. In a label distribution vector, the category corresponding to the maximum value in the probability distribution was identified as the label estimated by PENCIL. If this label was the same as the noise-free ground-truth label, we say it was correct. The results for 70% symmetric and 30% asymmetric noise on CIFAR-10 are shown in Fig. 2 and Fig. 3, respectively. We can observe that PENCIL effectively and stably estimated correct labels for most examples even with high noise rates. For example, with 70% symmetric noise rate, originally only about 16000 labels were correct, but after PENCIL's learning process there are about 39000 correct labels.

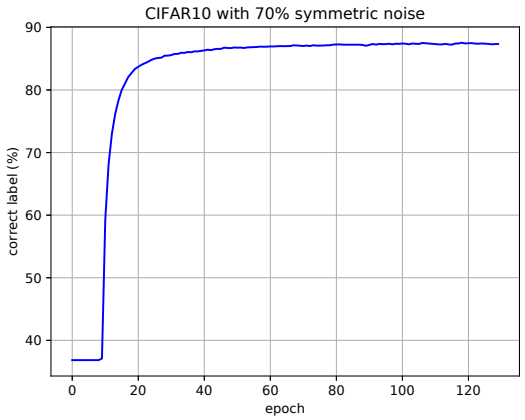

Figure 3. Correct labels on CIFAR-10 with 70% symmetric noise.

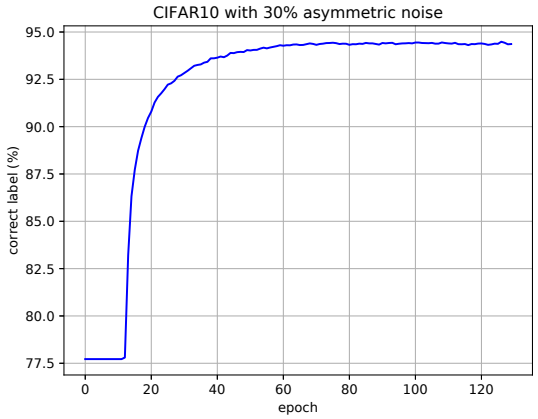

Figure 4. Correct labels on CIFAR-10 with 30% asymmetric noise.

4.5 Analyzing the Two Classification Losses

In this section, we compared the performance of KL- and inverse KL-divergence in PENCIL using the CIFAR-10 dataset. We observed the value and the gradient of $\boldsymbol{y}^d$ with original and inverse KL-divergence respectively and analysed the trends of them.

First of all, we observed the value of $\boldsymbol{y}_d$ directly. We turn the vector $\boldsymbol{y}_d$ into a label by finding its maximum value. As shown in Figure 3 and Figure 4, we can see that the labels are corrected successfully when we used the inverse KL-divergence. However, when we used the original KL-divergence following the same setting, we see that the number of correct labels is unchanged and the curve of it is a horizontal line (figures not shown). Therefore, the original KL-divergence is not suitable for noise correction in PENCIL.

Next, we randomly selected some images from CIFAR-10 with 30% symmetric noise to observe the gradient of $\boldsymbol{y}^d$. We consider two cases: when the original label is wrong (Figure 5) or correct (Figure 6).

4.5.1 When the Original Label Is Wrong

In Figure 5, the blue curve represents the component in the gradient of $\boldsymbol{y}_d$ corresponding to the original label (which is incorrect), and the orange curve is for the correct label. The

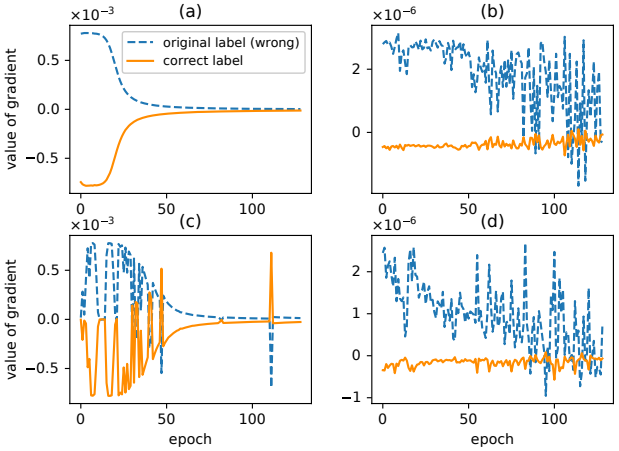

Figure 5. The component in the gradient of $\boldsymbol{y}^d$ corresponding to the correct label (solid orange curve) and the original label (blue dashed curve) when the original label is wrong. Figure (a) and (c) used the inverse KL-divergence. Figure (b) and (d) used the original KL-divergence. The top and bottom rows represent two different input images.

left two figures (a) and (c) show the results of two different input images with inverse KL-divergence. In these figures, we can see the blue curves are almost positive while the orange curves are almost negative. Therefore the labels are continuously corrected in PENCIL. We can see that both the correct and the original labels are successfully updated towards their desired values during training. In figures (b) and (d), which used the original KL-divergence, the magnitudes of the gradient (about 10^{-6}) are much smaller than those in the left two figures (about 10^{-3}). It is too small to correct the noisy labels. It's obvious that the original KL-divergence is not suitable in correcting label noise.

4.5.2 When the Original Label Is Correct

In this case we want the labels to remain unchanged. In other words, we want the gradient of $\boldsymbol{y}^d$ as small as possible. As shown in Figure 6, we can see the magnitudes of the left two figures are much smaller than those in the right two figures. Therefore the performance of the inverse KL-divergence is better than the original KL-divergence again.

Through the above two cases, we can conclude that the original KL-divergence is not suitable in our PENCIL framework, but our proposed inverse KL-divergence is.

4.6 Repetitive Training on CIFAR-10

We tested PENCIL with repetitive training on CIFAR-10 and reported results on the teest set of each iteration in the repetitive training process. All the settings of baseline is the same as the description in Section 4.2. In repetitive training, the learning rate and the λ will be reduced slightly according to the number of iterations.

The results are shown in Table 5. All the results are the *last*-epoch accuracy. We just run the whole repetitive training process once, therefore there are some small differences on the values of accuracy between Table 5, Table 3 and Table 4.

As shown in Table 5, we can see the accuracy always achieve best performance after repeating twice or thrice.

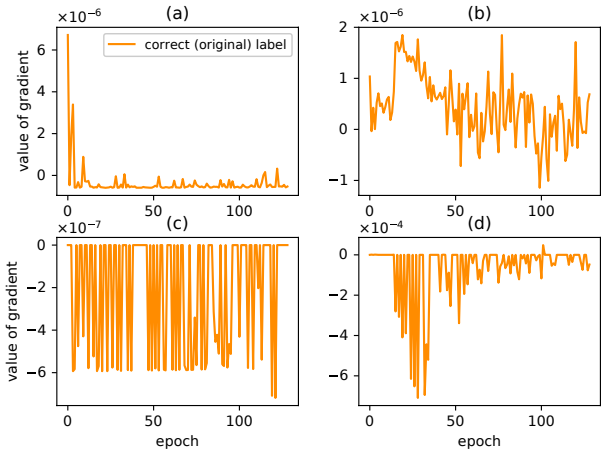

Figure 6. The component in the gradient of $\boldsymbol{y}^d$ corresponding to the correct (orignial) label. Figure (a) and (c) used the inverse KL-divergence. Figure (b) and (d) used the original KL-divergence. The top and bottom rows represent two different images.

Table 5
Test set accuracy on CIFAR-10 with symmetric and asymmetric noise. We reported results of each iteration in the repetitive training process.

Symmetric Noise					
#	noise rate (%)	10	30	50	70
1	baseline	93.50	92.13	90.46	86.95
2	repeat once	93.73	92.37	91.46	87.98
3	repeat twice	**94.08**	93.04	91.64	88.91
4	repeat thrice	93.78	**93.28**	**92.21**	**89.40**
Asymmetric Noise					
#	noise rate (%)	10	20	30	40
6	baseline	93.50	92.64	91.46	91.02
7	repeat once	93.22	92.90	91.42	**91.68**
8	repeat twice	**93.58**	92.97	**92.15**	91.64
9	repeat thrice	93.55	**93.35**	92.01	91.53

The accuracy can be sorted as: baseline $<$ repeat once $<$ repeat twice $\approx$ repeat thrice. In addition, the performance of repetitive training with high level noise is close to or even better than baseline with low level noise (e.g., the accuracy of repeating once with 40% asymmetric noise is 91.68%, which is better than the baseline with 30% asymmetric noise of 91.46%).

4.7 Experiments on CUB-200

We performed additional experiments on CUB-200 with different hyperparameters α and λ. This dataset is generally considered to contain no or only few noisy labels. Therefore, we use it to further test the robustness of PENCIL on problems not affected by noisy labels.

The results are listed in Table 6. Row $\#1$ is the baseline (classic method) and rows $\#2$ to $\#7$ are PENCIL results. For a wide range of α and λ values, PENCIL consistently exhibited competitive results (i.e., without obvious degradation). Furthermore, we observed the final label distributions, and the maximum values of all label distributions are correct (i.e., same as the correct labels). This observation shows that PENCIL works robustly in clean datasets, too.

In the settings of rows $\#4$ to $\#7$, PENCIL achieved higher accuracy than the baseline. In particular, row $\#4$ is 0.71% higher. A small percentage of label noise may exist in this dataset [33]. Our hypothesis is that by replacing the original one-hot label with probabilistic modeling in PENCIL, we obtained better robustness and consequently a small edge in accuracy.

Table 6
Test accuracy on CUB-200 with different hyperparameters. The accuracy of PENCIL does not decline in standard datasets with clean labels.

#	method		Test Accuracy (%)
1	Cross Entropy Loss		81.93
	PENCIL		
	λ	α	
2	1000	0	81.91
3	2000	0	81.84
4	3000	0	**82.64**
5	1000	0.1	82.09
6	2000	0.1	82.21
7	3000	0.1	82.22

Table 7
Performance of networks with different sizes with or without the proposed attention structure on the MS-COCO dataset. We selected efficientnet-b0 to -b5 [36] as our backbone networks to show the effectiveness of our attention structure comprehensively.

#	backbone network	mAP	
		w/o attention	w/ attention
1	efficientnet-b0	60.26	**68.06**
2	efficientnet-b1	62.48	**69.56**
3	efficientnet-b2	65.61	**71.09**
4	efficientnet-b3	68.62	**73.48**
5	efficientnet-b4	70.97	**76.33**
6	efficientnet-b5	72.21	**78.77**

4.8 Experiments on MS-COCO

Next, we show the effectiveness of our proposed attention structure in networks with different sizes on the MS-COCO dataset. We compared the performance of networks with and without the proposed attention structure. Efficientnet is suitable for our evaluations because it has versions with different sizes to test our proposed attention structure comprehensively.

The results are shown in Table 7. In all the rows, the performance of networks with the proposed attention structure outperforms those without it. Even on efficientnet-b5 which is a large scale network, our attention structure still achieved 6.56 percentage points mAP higher than baseline network.

In Table 7, the result of the network with attention structure in row $\#1$ (68.06) is better than the networks without our attention structure in rows $\#2$ (62.48) and $\#3$ (65.61), and even close to the network without the attention structure in row $\#4$ (68.62). However, the input image's size of efficientnet-b0 (224) is smaller than efficientnet-b1 (240), b2 (260) and b3 (300). The same holds for their model sizes. That means the computing cost of the former one is much less than the latter three. In comparison, the extra computing cost of our attention structure is small.

4.9 Experiments on a Subset of Open Images

Then we tested our PENCIL framework on a subset of Open Images, which is a large scale real-world multi-label

Table 8
Performance of backbone network with or without the proposed attention structure and PENCIL with attention structure on a subset of the Open Images dataset.

#	method	mAP
1	Baseline	62.55
2	Baseline w/ attention structure	75.11
3	PENCIL w/ attention structure	**77.13**

Table 9
Test accuracy on the Clothing1M dataset. Rows $\#1$ and $\#2$ were quoted from [12] and $\#3$ was quoted from [17]. These baseline methods used the complete Clothing1M training data, but our method only used a small pseudo-balanced subset (i.e., balanced in terms of noisy labels). Our method achieved state-of-the-art result in this real-world dataset.

#	method	Test Accuracy (%)
1	Cross Entropy Loss	68.94
2	Forward [12]	69.84
3	Tanaka *et al.* [17]	72.16
4	PENCIL	**73.49**

dataset with noisy labels. Because PENCIL corrects the noisy labels based on the backbone network, we need a backbone network with a reasonable starting point. Therefore we need to combine both PENCIL and the attention structure.

The results are shown in Table 8. Same as the results on MS-COCO, the performance of baseline with attention structure is much better than baseline without attention structure. Therefore our attention structure is still effective on the multi-label dataset with noisy labels. In row $\#3$, when combining PENCIL and the attention structure, we obtain the best performance. That means our proposed PENCIL framework is also effective on multi-label tasks.

4.10 Experiments on Clothing1M

Finally, we tested PENCIL on Clothing1M, which is a real-world noisy label dataset. It includes a lot of unknown structure (asymmetric) noise.

The results are shown in Table 9. All results are $best$ test accuracy. Rows $\#1$ and $\#2$ were quoted from [12], and row $\#3$ was reported in [17]. Although these baseline models were trained on the whole Clothing1M training set, our PENCIL used a randomly sampled pseudo-balanced subset, including about 260k images. The backbone network was ResNet-50 for all methods.

In Table 9, only noisy labeled examples were used (i.e., without using the clean training subset). The Forward [12] method required the ground-truth noise transition matrix, which is not available. Hence, it used an estimated matrix instead. The Tanaka *et al.* [17] method used the distribution of noisy labels to relieve the imbalanced problem. In our PENCIL method, we did not use any extra prior information. PENCIL achieved 1.33% higher accuracy than that of Tanaka *et al.* [17], 3.65% higher than Forward [12] and 4.55% than cross entropy.

5 Conclusion

We proposed a framework named PENCIL to solve the noisy label recognition problem in deep leaning. PENCIL adopted label probability distributions to supervise network learning and to update these distributions through back-propagation end-to-end in every epoch. We proposed an inverse KL-loss, which is different from previous methods but is robust for noisy label handling, then we show that the inverse KL-loss is indeed more suitable than the original KL-loss. The proposed PENCIL framework is end-to-end and independent of the backbone network structure, thus it is easy to deploy. Then we find that repetitive training of PENCIL can achieve better performance. Finally we extend PENCIL to multi-label classification tasks with our proposed attention structure.

We tested PENCIL with synthetic label noise on CIFAR-100 and CIFAR-10 with different noise types and noise rates, and outperformed current state-of-the-art methods by large margins. On CIFAR-10, we also show the effectiveness of repetitive training and the inverse KL-loss is more suitable than the original KL-loss. We also experimented on CUB-200, which is considered to be noise free. The results show that PENCIL is robust for different datasets and hyperparameters. Then we evaluated our proposed attention structure on multi-label dataset MS-COCO with different networks sizes, which show our attention structure is effective in multi-label classification. Next, We tested PENCIL with attention structure on a subset of Open Images which is a large scale real-world multi-label dataset with noisy labels. The results show the effectiveness of our PENCIL framework with attention structure on multi-label dataset with noisy labels. Lastly, we tested PENCIL on the real-world large scale single label noise dataset Clothing1M. On this dataset, we achieved 1.33% higher accuracy than previous state-of-the-art.

References

[1] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.

[2] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[3] R. Fergus, F. Li, P. Perona, and A. Zisserman, "Learning object categories from Internet image searches," *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1453–1466, 2010.

[4] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The unreasonable effectiveness of noisy data for fine-grained recognition," in *ECCV*, ser. LNCS, vol. 9907. Springer, 2016, pp. 301–320.

[5] F. Schroff, A. Criminisi, and A. Zisserman, "Harvesting image databases from the web," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 4, pp. 754–766, 2011.

[6] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *CVPR*, 2015, pp. 2691–2699.

[7] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *ICCV*, 2017, pp. 843–852.

[8] D. Angluin and P. D. Laird, "Learning from noisy examples," *Machine Learning*, vol. 2, no. 4, pp. 343–370, 1988.

[9] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[10] 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.

[11] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," *arXiv preprint arXiv:1406.2080*, 2014.

[12] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *CVPR*, 2017, pp. 1944–1952.

[13] J. Yao, J. Wang, I. W. Tsang, Y. Zhang, J. Sun, C. Zhang, and R. Zhang, "Deep learning from noisy image labels with quality embedding," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1909–1922, 2018.

[14] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR*, 2017.

[15] A. Vahdat, "Toward robustness against label noise in training deep discriminative neural networks," in *NIPS*, 2017, pp. 5601–5610.

[16] B. Gao, C. Xing, C. Xie, J. Wu, and X. Geng, "Deep label distribution learning with label ambiguity," *IEEE Trans. Image Processing*, vol. 26, no. 6, pp. 2825–2838, 2017.

[17] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *CVPR*, 2018, pp. 5552–5560.

[18] K. Yi and J. Wu, "Probabilistic end-to-end noise correction for learning with noisy labels," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 7017–7025.

[19] J. Larsen, L. N. Andersen, M. Hintz-Madsen, and L. K. Hansen, "Design of robust neural network classifiers," in *ICASSP*, 1998, pp. 1205–1208.

[20] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 5, pp. 845–869, 2014.

[21] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise," *arXiv preprint arXiv:1705.10694*, 2017.

[22] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *J. Artif. Intell. Res.*, vol. 11, pp. 131–167, 1999.

[23] I. Guyon, N. Matic, and V. Vapnik, "Discovering informative patterns and data cleaning," in *KDD*, 1996, pp. 181–203.

[24] K. Lee, X. He, L. Zhang, and L. Yang, "CleanNet: Transfer learning for scalable image classifier training with label noise," in *CVPR*, 2018, pp. 5447–5456.

[25] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S. Xia, S. N. R. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *ICML*, 2018, pp. 3355–3364.

[26] 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, pp. 8688–8696.

[27] A. Ghosh, H. Kumar, and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," in *AAAI*, 2017, pp. 1919–1925.

[28] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *NIPS*, 2018.

[29] G.-H. Wang and J. Wu, "Repetitive reprediction deep decipher for semi-supervised learning," in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[30] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[32] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv:1811.00982*, 2018.

[33] M. J. Wilber, I. S. Kwak, D. J. Kriegman, and S. J. Belongie, "Learning concept embeddings with combined human-machine expertise," in *ICCV*, 2015, pp. 981–989.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[35] ——, "Identity mappings in deep residual networks," in *ECCV*, ser. LNCS, vol. 9908. Springer, 2016, pp. 630–645.

[36] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.