KNN-enhanced Deep Learning Against Noisy Labels

Shuyu Kong[*1], You Li [†1], Jia Wang[‡2], Amin Rezaei[§3], and Hai Zhou[¶1]

[1]*Department of Electrical and Computer Engineering, Northwestern University*
[2]*Department of Electrical and Computer Engineering, Illinois Institute of Technology*
[3]*Department of Computer Engineering and Computer Science, California State University Long Beach*

Abstract

Supervised learning on Deep Neural Networks (DNNs) is data hungry. Optimizing performance of DNN in the presence of noisy labels has become of paramount importance since collecting a large dataset will usually bring in noisy labels. Inspired by the robustness of K-Nearest Neighbors (KNN) against data noise, in this work, we propose to apply deep KNN for label cleanup. Our approach leverages DNNs for feature extraction and KNN for ground-truth label inference. We iteratively train the neural network and update labels to simultaneously proceed towards higher label recovery rate and better classification performance. Experiment results show that under the same setting, our approach outperforms existing label correction methods and achieves better accuracy on multiple datasets, e.g., 76.78% on Clothing1M dataset.

1 Introduction

Deep Neural Networks (DNNs) have achieved remarkable success in various applications including computer vision, speech recognition, and robotics. Supervised learning on DNNs is data hungry. Obtaining a large dataset with labels at an affordable cost is usually done by crowdsourcing [33, 36] and web query [25, 34]. Each of those would inevitably introduce a significant amount of noisy labels. On the other hand, DNNs are prone to overfit noisy training data [36, 1], and their generalization performance is

downgraded as a result.

To resist noisy labels in training DNNs, numerous methods have been proposed, including robust loss formulation [22, 9, 31], curriculum learning [10] and label correction [35, 28]. In this paper, we focus on label correction approach which alternatively sanitizes noisy labels and improves the model performance. Previous work leverages prediction from DNN itself to infer the ground truth labels [35, 28]. However, such prediction is likely to be poisoned by the noise in the training dataset. Therefore, we are motivated to seek for a more robust label correction approach.

In this paper, we leverage the deep K-Nearest Neighbors (KNN) algorithm to facilitate learning with noises. The KNN algorithm assumes that similar things exist in close proximity. KNN is a favorable classification approach when no prior knowledge on sample distribution are available, and has shown robustness against adversarial examples [26, 20, 30]. Our approach is based on a key observation that during the learning phase, useful features are learnt in the intermediate layers despite the presence of corrupted labels in the dataset. We propose to use those features to discover similarity among samples. Even though the final labels are different for two samples belonging to same category, their features share high similarity.

Overall, we present a framework that iteratively applies deep KNN to infer ground truth labels and retrains the neural network with the predicted labels, thus simultaneously making progress toward higher label recovery and better classification performance. It is a generalized framework that does not require an estimation of noise transition distributions or a clean dataset for reference. We also propose two KNN label correction algorithms.

---
[*]shuyukong2020@u.northwestern.edu
[†]you.li@u.northwestern.edu
[‡]jwang34@iit.edu
[§]me@aminrezaei.com
[¶]haizhou@northwestern.edu

1

The first one, IterKNN, uses all the labels to infer ground truth labels. The second one, SelKNN, selects a certain amount of clean examples as reference for KNN ground truth inference. The selection principle is to find samples with small cumulative normalized loss because those samples are more likely to have correct labels. We empirically show that our approach achieves state-of-art performance.

The contributions of this paper are as follows:

- To our best knowledge, we are the first to apply deep KNN for label correction in corrupted training dataset. The features are extracted from intermediate layers of neural network. To further mitigate the impact of noisy labels, we propose a loss ranking approach to select samples with high confidence to be labelled correctly as reference for KNN prediction.
- We explore the benefits of iterative retraining after label correction. We show that iterative retraining can help neural network escape from over-fitting and discover more corrupted labels, thus achieves better performance.
- We conduct extensive experiments to demonstrate the robustness of deep KNN against label noise. We found out that even though the final prediction is corrupted by the noise in training dataset, CNN can still learn robust and useful features in deep layers to facilitate KNN for ground-truth label inference. We also provide insights on how deep feature is better than both of the shallow features and final logits with regard to noisy label correction..

2 Related Work

2.1 Generalization of Deep Neural Networks

Zhang *et al.* [36] showed that deep neural networks have the capacity to memorize completely random labels, but may result in poor generalization. This indicates that label corruption in training dataset has a negative impact on DNN performance. Other studies [16] demonstrated that DNN tends to learn clean labels first and that overfitting to corrupted labels requires to stray far from initialization. Thus early stop is a simple but effective method to resist noisy labels.

2.2 Semi-Supervised Deep Learning

The goal of semi-supervised learning is to learn from partially labelled dataset. With the development of DNNs, researchers have studied how DNN can learn in semi-supervised setting. One methodology is to add an unsupervised loss term as regularizer to force mutual exclusiveness of different classes [29, 24]. Another popular approach in semi-supervised deep learning is to assign pseudo-labels to unlabeled examples. The pseudo-labeled data are trained in the supervised fashion. Ahmet *et al.* [13] have proposed transductive label propagation using nearest neighbor graph. Our approach can be viewed as label propagation from clean examples to noisy examples. However, our problem is more challenging because we do not assume that clean examples are provided.

2.3 Noisy Label Learning

Loss correction approach is widely studied in training with noisy labelled data. Forward backward loss correction [22] directly leverages the noisy transition matrix T to modify the Cross Entropy (CE) loss. But in practice T is usually not given. Other studies attempt to estimate the noisy transition matrix by modelling it with a fully connected layer [27, 15], or use bootstrapping to avoid direct noise modelling [23]. Recently, Wang *et al.* [31] have proposed to combine CE with Reverse Cross Entropy (RCE) and demonstrated that Symmetric Learning (SL) can avoid overfitting noisy data.

The field of Curriculum Learning (CL), which is motivated by the idea of a curriculum in human learning, attempts at imposing some structure on the training set. Such structure essentially relies on a notion of "easy" and "hard" examples, and utilizes this distinction in order to teach the learner how to generalize easier examples before harder ones [10]. Empirically, the use of CL has been shown to accelerate and improve the learning process [3] and noise robustness [4].

Noisy label detection and filtering is another approach to mitigate data noise [12]. This is based on the fact that removing corrupted data can improve model performance. However, the hard samples may be confused with the noisy ones. As a consequence, certain amount of hard samples are filtered out together with noisy samples. The resultant clean dataset contains most of the easy samples,

which will make the classifier less generalizable.

Other methods attempt to correct noisy labels. Tanaka *et al.* [28] have formulated the noisy label learning problem as a joint optimization problem where model parameters and data sample labels are optimized alternatively. Yi *et al.* [35] also have targeted the optimization problem by concurrently updating noisy labels and model parameter. Both methods require the prior knowledge over label distribution in order to be effective. However, in practice it is difficult to obtain such prior knowledge.

2.4 Robust K-Nearest Neighbors

KNN is a favorable classification approach when no prior knowledge on sample distribution is available, and is believed to be resistant to label noise. Gao *et al.* [8] have provided theoretical analysis on the robustness of KNN against noisy labels for binary classification. For symmetric noise where the labels of each class have equal probability to be flipped to the other, they showed that KNN can reach the same consistent rate as in the noise free setting. As for asymmetric noise, KNN is still robust since most samples can be correctly classified. They then derived a strategy to deal with noisy label classification with KNN. Specifically, their method corrects the labels of "totally misled" samples. How the labels should be corrected is depending on the estimations of noise proportions [17, 19].

Another strategy to resist label noise with KNN is to emphasize the more trustworthy information. Parvin *et al.* [21] assign weights to every sample based on its validity. On the other hand, it is shown that to assign weights to features can also improve robustness and accuracy for KNN [14].

An earlier work [2] proposes to use KNN for selecting clean examples. It shows that a simple KNN filtering approach on the logit layer of a preliminary model can remove mislabeled training data and help produce more accurate models. This work is most closely related to ours. But we want to highlight several difference between the two works and address our novel contribution on combining deep KNN inference with noisy learning. Firstly, instead of only filtering out suspicious examples, our work attempts to use KNN to correct corrupted labels so that the noisy examples are recycled to augment the training dataset, thus boosting the generalization performance of the final model. Besides, we also study the effect of selecting different layer as features on the KNN correction performance rather than restricting choice of only extracting the last logit layer. Last but not least, we discover the disadvantage of adopting the whole training dataset as KNN reference (as done in IterKNN) when the noisy level is high. Therefore, we propose a clean example selection strategy (as done in SelKNN) based on ranking the cumulative training loss with the hope that noisy labels will be mostly corrected by clean labels as reference for KNN classification.

3 Main Approach

3.1 Preliminaries and Problem Statement

We are targeting a multiclass classification problem with label noise. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the image space and label space, respectively. Let $\tilde{D}_n = \{(\boldsymbol{x}_1, \tilde{y_1}), \cdots, (\boldsymbol{x}_n, \tilde{y_n})\}$ be the noisy training dataset where $x_i \in \mathcal{R}^d$ is a d-dimensional vector and $\tilde{y}_i \in \mathcal{Y} = \{1, ..., C\}$. True labels $y_i, \cdots, y_n$ are not observable. There exists a hidden noise model T, while $t_i = P(\tilde{y}_i | x_i, y_i)$ representing the probability the true label y_i of instance x_i is flipped to $\tilde{y}_i$. Note if T is only class-dependent, we will have $t_i = P(\tilde{y}_i | y_i)$, which is assumed in some existing work [22]. Consequently, the overall label error rate on the dataset due to noise is defined as $Pr(\tilde{y}_i \neq y_i)$.

When the dataset is clean, the learning problem can be formulated into an optimization problem as follows:

$$\theta^* = \operatorname*{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^{n} J(y_i, \mathcal{F}(x_i, \theta)), \qquad (1)$$

where $\mathcal{F}$ is a classifier with parameters θ and J is the target loss function. The objective is to find optimal parameters θ^* such that average loss is minimized. However, when labels are corrupted, we can not obtain the above formulation since ground truth labels $y_i, \cdots, y_n$ are unknown. Therefore, only corrupted labels can be used in the following optimization problem:

$$\tilde{\theta}^* = \operatorname*{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^{n} J(\tilde{y}_i, \mathcal{F}(x_i, \theta)). \qquad (2)$$
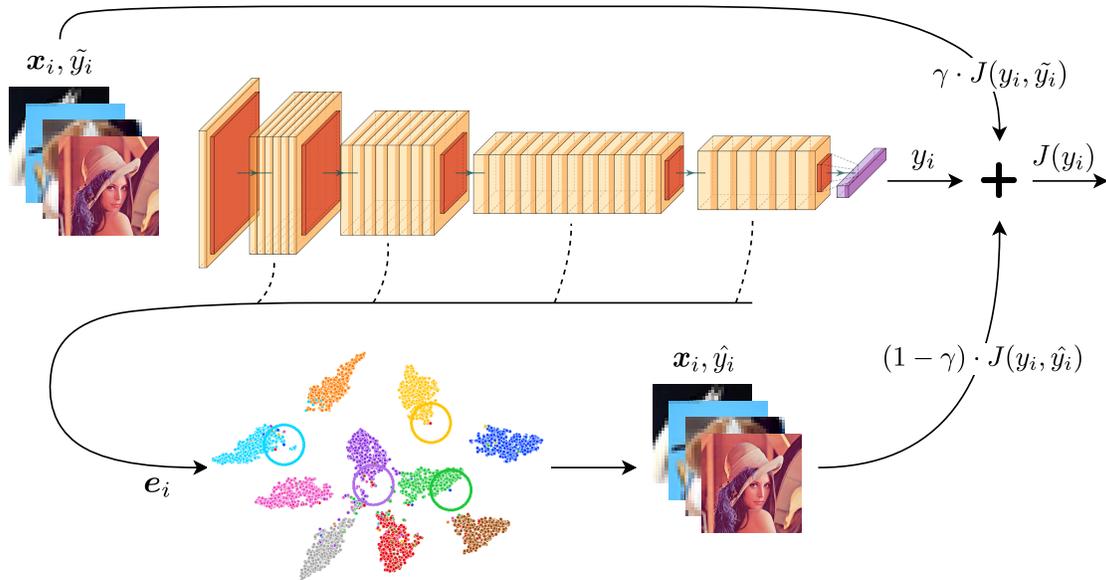
Figure 1: The overall architecture of IterKNN. The deep neural network extracts embeddings of samples with intermediate layers. After every training episode, KNN classifier corrects labels in the reference dataset based on the embeddings of the samples. The deep neural network is trained on a hybrid loss function, which is comprised of the loss on the original labels and the loss on KNN predicted labels.

$\tilde{\theta}^*$ is only a sub-optimal solution to Equation 1. To approach the real optimal solution θ, we attempt to detect and correct the corrupted labels based on the following fact:

$$\theta^* = \lim_{y \to \tilde{y}} \tilde{\theta}^*. \qquad (3)$$

It is also important to develop a method agnostic to T, that is, we neither assume any structure nor require any prior knowledge of noise model T.

3.2 Deep k-Nearest Neighbor Label Correction

k-Nearest Neighbor is a widely used non-parametric classification approach. It predicts the label of input by finding a total of k nearest neighbors and then taking a majority voting of the labels of those neighbors:

$$\eta_k(\boldsymbol{x}) = \operatorname*{argmax}_y \sum_i^n 1 \cdot (y = y_i, \boldsymbol{x} \in N_k(\boldsymbol{x}_i)), \quad (4)$$

in which $\eta_k(\boldsymbol{x})$ is the label prediction for sample $\boldsymbol{x}$, and $N_k(\boldsymbol{x})$ is some distance matrix.

Inspired by the robustness of KNN against noisy dataset [8], we propose to apply KNN on label correction. It requires to define a distance metric to measure the similarity between image samples. One straightforward solution is to compute the total pixel-wise difference between pairs of images. Yet such a metric is weak even toward a very small transformation. The most popular approach is to measure the distance in feature space. Consequently, one critical question is what kind of features we should use to represent each image.

Traditional offline feature extraction approaches like SIFT [18] and HOG [7] can resist transformations, but may not preserve enough class-related information for classification. As deep learning becomes dominant in vision-related problems, deep feature extraction attracts more and more attention and shows state-of-art performance in downstream tasks. In this paper, we use the deep representation learned by the neural network as features. More specifically, we take the intermediate layers from the neural network being trained as feature extractors. Even though the training dataset is corrupted, the deep layers of DNN can still extract useful features (em-

4

beddings). We will empirically show and validate such a result in Sec. 4.3. Moreover, as noisy labels get corrected by KNN in every episode, the DNN can be trained on more accurate data and thus produces features with higher quality. The better features can further facilitate KNN for more accurate label correction. Such a process forms a benign cycle, and benefits both label correction and neural network training as the result.

Algorithm 1 Iterative Training and Correction Algorithm

Input: the noisy training set $\tilde{S}_n$, initial network model θ_0, k, number of training episodes M, number of training epochs T, initial learning rate ϵ_0, loss coefficient γ.

1: Initialize KNN corrected set $\hat{S}_n$ to noisy set $\tilde{S}_n$
2: **for** $m \leftarrow 1, M$ **do**
3:     (re)Initialize θ to θ_0
4:     (re)Initialize ϵ to ϵ_0
5:     **for** $t \leftarrow 1, T$ **do**
6:         Update θ at training rate ϵ with loss function:
            $(1 - \gamma) \cdot J(\theta; \boldsymbol{x}, \hat{y}) + \gamma \cdot J(\theta; \boldsymbol{x}, \tilde{y})$
7:         Update learning rate ϵ accordingly
8:     **end**
9:     $\hat{S}_n \leftarrow$ label_correction($\hat{S}_n$) // either IterKNN or SelKNN
10:     Decay loss coefficient γ
11: **end**
12: $S_n^* \leftarrow \hat{S}_n$
13: Train θ by training neural network on $\hat{S}_n$
    Output: corrected set S_n^*, trained network model θ.

3.3 Iterative Error Label Correction

After one training episode is finished, we can extract deep embeddings of all the samples in the training dataset and perform KNN label correction. Then we move to the next episode with the updated labels. At this point, the neural network may already overfit the noisy labels in the training dataset. As we have mentioned, the quality of embedding extraction by the neural network and the quality of label correction by KNN are correlated. Being trapped in overfitting to noisy labels will adversely affect both qualities.

To overcome overfitting, we regularly reinitialize the neural network model at the beginning of every episode.

This will result in a more accurate neural network model in every episode, as long as the label accuracy is improved by label correction. The overall procedure is shown in Algorithm 1.

3.4 Selective Label Correction

In this paper, we propose two KNN label correction approaches, IterKNN and SelKNN. In IterKNN, the features of all the samples along with their noisy labels are used as reference for KNN label inference as shown in Algorithm 2. However, datasets for real applications are typically very large. That makes it very expensive to launch KNN classification for every single instance across the whole training dataset. In that sense, a better approach is to selectively choose reference samples for KNN. The most naive selection heuristic is random sampling in the dataset. On the other hand, since the dataset itself is corrupted, the KNN sample set could contain a large portion of samples which have corrupted labels.

Our objective is to ensure the KNN sample set contains as many clean samples as possible to make it more trustworthy. One simple but effective heuristic to filter clean samples is to utilize the loss information. In general, clean samples are easier to learn and can be learned by the neural network faster. Reversely, noisy labels are more difficult and can only be memorized in the later stage of training [5]. Based on that, we believe cumulative normalized loss is a good factor to determine the likelihood of samples having corrupted labels.

Accordingly, we propose selective label correction algorithm (SelKNN) as shown in Algorithm 3. We track and accumulate the normalized loss of every image in all previous epochs. At the end of an episode, we rank the cumulative normalized loss for all samples and pick M samples with lowest loss from each class c as the reference of KNN classifier:

$$B_c((\boldsymbol{x}, \hat{y}), l) = \{(\boldsymbol{x}, \hat{y}) \in \hat{S}_n | J(\theta; \boldsymbol{x}, \hat{y}) \leq l_c, \hat{y} \in c\}, \tag{5}$$

where $(\hat{\boldsymbol{x}}, \hat{y})$ are drawn from the current $\hat{S}_n$, y is the prediction of the current $\mathcal{F}(x, \theta)$, and l_c is the infimum of cumulative loss that class c has M samples with cumulative losses less than or equal to it:

$$l_c = \inf\{l : |(\boldsymbol{x}, \hat{y})| = M, J(\theta; \boldsymbol{x}, \hat{y}) \leq l\}. \tag{6}$$

5

The labels of those samples in B_c are kept unchanged, while the labels of the remaining samples will be corrected and updated with KNN according to the samples in B_c.

Algorithm 2 KNN Label Correction (**IterKNN**)

1: **procedure** label_correction($\hat{D}_n$)
2:    $\hat{D}_n{}' \leftarrow \{\}$
3:    **for** $(\boldsymbol{x}_i, \hat{y}_i)$ **in** $\hat{D}_n$ **do**
4:       Simulate embedding $\boldsymbol{e}_i$ with $\mathcal{F}(\boldsymbol{x}_i, \theta)$
5:       $\hat{y}_i{}' \leftarrow argmax_y \sum_i^n 1 \cdot (y = \hat{y}_i, \boldsymbol{e} \in N_k(\boldsymbol{e}_i))$
6:       $\hat{D}_n{}' \leftarrow \hat{D}_n{}' \cup (\boldsymbol{x}_i, \hat{y}_i{}')$
7:    **end**
8:    **return** $\hat{D}_n{}'$
9: **end procedure**

Algorithm 3 Selective KNN Label Correction (**SelKNN**)

1: **procedure** selective_label_correction($\hat{D}_n$)
2:    $B \leftarrow \{\}$
3:    **for** c **in** $\mathcal{Y}$ **do**
4:       Find the loss threshold l_c of class c according to Equation 6
5:       Select reference samples B_c for KNN according to Equation 5
6:       $B \leftarrow B \cup B_c$
7:    **end**
8:    $\hat{D}_n{}' \leftarrow B$
9:    **for** $(\boldsymbol{x}_i, \hat{y}_i)$ **in** $\hat{D}_n \setminus B$ **do**
10:       Simulate embedding $\boldsymbol{e}_i$ with $\mathcal{F}(\boldsymbol{x}_i, \theta)$
11:       $\hat{y}_i{}' \leftarrow argmax_y \sum_i^n 1 \cdot (y = \hat{y}_i, \boldsymbol{e} \in N_{k,B}(\boldsymbol{e}_i))$
12:       $\hat{D}_n{}' \leftarrow \hat{D}_n{}' \cup (\boldsymbol{x}_i, \hat{y}_i{}')$
13:    **end**
14:    **return** $\hat{D}_n{}'$
15: **end procedure**

3.5 Design of Loss Function

The DNN part in the framework keeps being updated with regard to the loss function. If only the corrected labels are used in training, the network will reach self-convergence quickly. It is because of the nature of KNN: KNN will correct the noisy or hard samples, and result in a dataset that is easier for the network to learn. To resolve this quick self-convergence problem, we always keep a copy of the original noisy dataset along with the corrected dataset, which is updated after every episode. We implement a total loss that is a convex combination of the losses on the two datasets:

$$J_{train} = (1-\gamma) \cdot J(\mathcal{F}(\boldsymbol{x}, \theta), \hat{y}) + \gamma \cdot J(\mathcal{F}(\boldsymbol{x}, \theta), \tilde{y}), \quad (7)$$

where γ is the weight coefficient for the total loss J_{train}, which is distributed between the original noisy dataset $\tilde{D}_n$ and the current KNN corrected dataset $\hat{D}_n$. Initially γ is set as 1.0, meaning in the first iteration we only consider the original noisy labels and ignore the KNN predicted labels. This makes sense since the DNN classifier has not been trained and no meaningful features can be obtained from it. To improve convergence of the algorithm, we decay γ by σ for every training iteration after the current labels are updated. As the episode index grows, our training loss tends to focus on the KNN predicted labels. Eventually, when γ gets close to 0, the neural network distill itself and the algorithm will converge. We realize our final loss is similar to bootstrapping loss [23]. The difference is that bootstrapping uses prediction from classifier itself while we use prediction from KNN. We will empirically show KNN prediction is more robust in Sec. 4.3. Overall architecture is illustrated in Figure 1.

4 Experiments and Evaluation

We conduct experiments and evaluate our approach on 3 popular synthetic datasets (MNIST, CIFAR-10, CIFAR-100), and 1 large-scale real life dataset (Clothing1M). All experiments are implemented using the PyTorch framework.

4.1 Noise Settings

We consider both symmetric and asymmetric label noises in our experiments.

Symmetric Noise: Given the noise level π, the label of every sample has a probability π to be flipped to another class uniformly at random:

$$\begin{cases} Pr(\tilde{y}_i = y_i) = 1 - \pi, \\ \forall j \neq i : Pr(\tilde{y}_i = y_j) = \dfrac{\pi}{|\mathcal{Y}| - 1}. \end{cases} (i, j \in \mathcal{Y}) \quad (8)$$

Asymmetric Noise: We use a similar configuration as discussed in [22]. We inject asymmetric label noises to mimic the part of the mistakes made between similar classes. For MNIST, we use the following class transitions: $7 \rightarrow 1$, $2 \rightarrow 7$, $5 \leftrightarrow 6$, and $3 \rightarrow 8$. For CIFAR-10, we have *truck→automobile*, *bird→airplane*, *cat↔dog*, and *deer→horse*. For CIFAR-100, we pair classes in a similar fashion.

4.2   Hyper-parameter Settings

We use Adam optimizer with weight decay of 1e-4 and initial learning rate of 0.001 for all experiments.

MNIST: We use a 5-layer CNN model with 4 convolution layers followed with a fully connected layer. Each training episode has 40 epochs and the learning rate is decreases by 10 times on epoch 20 and epoch 30. The batch size is 256.

CIFAR-10: We use Pre-act Resnet 32 [11]. Each episode of training has 120 epochs and the learning rate is decreased by 10 times on epoch 60 and epoch 90. For data augmentation, we use random crop, random horizontal flip, random affine and color jetter. The training batch size is 512.

CIFAR-100: We use Pre-act Resnet 56. Each episode of training has 120 epochs and the learning rate is decreased by 10 times on epoch 80 and epoch 120. Data augmentation and training batch size are same to those of CIFAR-10.

We compare our approach with the follow baselines:

Training with cross entropy: This is the most basic training setting, in which the neural network is trained directly with noisy labels using cross entropy loss. **Symmetric Loss [31]:** This approach uses a combination of cross entropy loss and reverse cross entropy loss. In that way it encourages learning hard samples without overfitting noisy labels. **Joint Optimization [28]:** This method optimizes the loss by updating network parameters and class labels alternatively. **PENCIL [35]:** This approach is similar to Joint Optimization. The only major difference is that differentiable psuedo labels are created and updated together with model parameters simultaneously. Both Joint Optimization and PENCIL add two extra losses, entropy loss and regularization loss, which require prior knowledge of the true class distribution.

All the above baselines are re-implemented according to their open-source codes with minor modifications to fit our settings. We always set k value to 100, using L2 distance metric, and use majority voting for label inference on both IterKNN and SelKNN. We perform in total 10 episodes of training together with label updating for every time of execution. Symmetric cross entropy loss is adopted to encourage learning hard examples. Weight coefficient γ is initially 1.0 and decayed by 1.2 every episode. For SelKNN, we pick up top $M\%$ of images from each class as KNN reference samples in order to update the labels of the the remaining $1 - M\%$ samples. M is set to 20 for first episode and is incremented by 10 every episode afterwards, until it reaches 100. We can generally increase M because more and more corrupted labels are cleaned up. When M reaches 100, SelKNN will reduce to IterKNN to facilitate further label correction. The comparison results are shown in Table 1. SelKNN achieves state-of-the-art performance on most noise settings, especially when noise rate is high while IterKNN has competitive performance when the noise rate is low. Another observation is the variance of the accuracy among multiple trials is low when the noise rate is low due to less randomness. When the noise rate increases, the variance of test accuracy obtained from SelKNN is least affected since it only pick highly possible clean samples to update labels. This result further validates the robustness of our SelKNN approach.

4.3   Robustness and Effectiveness of KNN

To demonstrate the robustness of KNN over DNN, we perform 5 types of 1-episode training on CIFAR-10 with 60% symmetric noise. Here we employ cross entropy loss to better differentiate the robustnesses of different predictions since it is relatively weak against label noise. We keep track of the ground-truth label inference accuracy as training proceeds. The results and details are illustrated in Fig. 3. All types of predictions overfit the data noise to some extent at the later stage of the episode. We let the DNN trained on noisy dataset to extract embeddings as well as predict labels for input images. When IterKNN and SelKNN are allowed to run classification on clean labels, their predictions are less subject to over-fitting than DNN prediction. This shows the DNN can still learn from noisy datasets, so that it derives similar features for
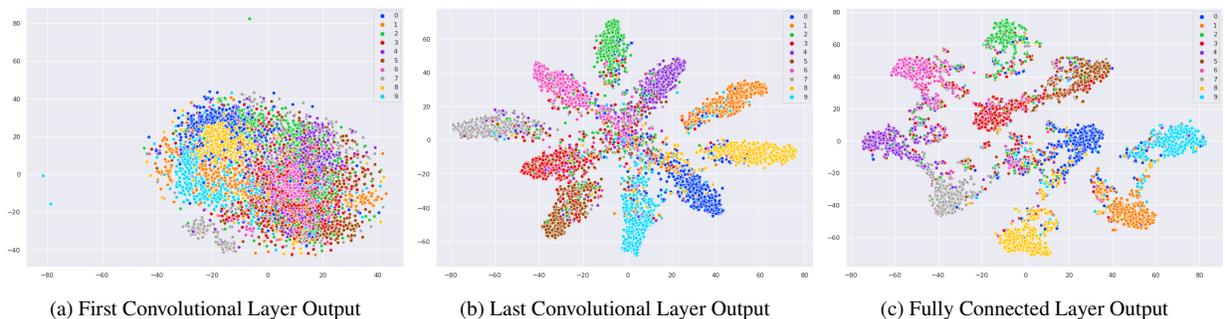
(a) First Convolutional Layer Output   (b) Last Convolutional Layer Output   (c) Fully Connected Layer Output

Figure 2: t-SNE 2D embeddings of outputs from different layers after 1 training episode. (CIFAR-10 with 60% symmetric noisy labels)

methods	symmetric noise rate				asymmetric noise rate		
	20%	40%	60%	80%	20%	30%	40%
MNIST							
CE	89.91±0.02	68.75±0.04	44.35±0.24	24.77±0.37	94.22±0.07	86.40±0.12	80.33±0.09
Joint Opt [28]	94.89±0.07	93.94±0.16	91.03±0.46	66.55±1.05	94.79±0.26	93.01±0.24	91.78±0.33
PENCIL [35]	95.34±0.21	93.89±0.72	93.06±0.53	70.25±0.86	96.22±0.18	94.38±0.21	90.15±0.42
SL [31]	**98.77**±**0.07**	97.65±0.19	95.27±0.35	68.04±0.33	98.61±0.21	97.11±0.81	96.15±0.69
IterKNN	98.33±0.27	96.35±0.22	94.33±0.32	68.81±0.49	98.65±0.11	97.27±0.15	94.18±0.41
SelKNN	98.25±0.12	**97.79**±**0.19**	**96.12**±**0.31**	**70.88**±**0.29**	**98.69**±**0.08**	**97.80**±**0.17**	**96.39**±**0.18**
CIFAR10							
CE	83.04±0.18	66.75±0.11	46.98±0.21	21.33±0.36	86.16±0.24	80.91±0.13	70.51±0.76
Joint Opt [28]	91.61±0.74	87.75±0.36	84.02±0.42	58.46±0.97	91.16±0.09	89.41±0.21	86.76±0.81
PENCIL [35]	**92.45**±**0.81**	88.32±0.49	84.42±0.98	59.02±0.72	90.21±0.16	89.36±0.32	87.59±0.45
SL [31]	87.89±0.07	84.36±0.09	79.75±0.18	55.36±0.42	87.44±0.22	85.24±0.25	80.32±0.29
IterKNN	89.43±0.14	87.34±0.23	80.19±0.21	56.84±0.38	90.29±0.22	89.28±0.18	87.15±0.33
SelKNN	91.99±0.44	**89.25**±**0.39**	**85.65**±**0.92**	**59.65**±**0.27**	**91.38**±**0.25**	**89.75**±**0.32**	**88.06**±**0.28**
CIFAR100							
CE	60.13±0.20	51.25±0.52	36.75±0.81	19.35±0.79	61.16±0.74	56.27±0.78	47.35±0.78
Joint Opt [28]	68.06±0.88	64.79±0.91	54.29±0.97	27.78±0.94	69.13±0.39	68.49±0.36	59.23±0.70
PENCIL [35]	71.14±0.34	68.75±0.62	56.02±0.66	26.33±0.58	**73.15**±**0.85**	71.02±0.63	61.48±0.94
SL [31]	64.85±0.29	59.42±0.34	45.85±0.46	22.81±0.79	65.19±0.21	63.37±0.45	60.01±0.65
IterKNN	69.15±0.30	62.48±0.91	53.36±0.47	28.93±0.98	69.92±0.42	68.13±0.37	62.14±0.39
SelKNN	**72.75**±**0.68**	**69.88**±**0.64**	**57.43**±**1.23**	**32.91**±**0.57**	72.03±0.26	**71.09**±**0.45**	**63.32**±**1.39**

Table 1: Accuracy(%) from the last training epoch on 3 datasets under different noise settings. We run 10 trials, report mean and std. Our approach achieves best results under most settings.

similar images, which validates the robustness of KNN based approach. On the other hand, when no ground-truth or clean labels are provided, both of IterKNN and SelKNN have slightly worse performance, but are still clearly better than the DNN prediction. All those observations lead to the following two conclusions: (1) our IterKNN and SelKNN should be more effective than those self-learning approaches using DNN prediction to recover labels [28]. (2) SelKNN can be improved with a better filtering method (*e.g.*, [12]) to select clean reference samples.

methods	symmetric noise		
	20%	40%	80%
Joint Opt	93.35±0.32	90.31±0.36	58.72±0.85
PENCIL	94.36±0.36	91.21±0.52	59.14±0.16
IterKNN	**94.47±0.25**	89.16±0.28	56.95±0.79
SelKNN	93.29±0.25	**91.32±0.22**	**59.94±0.52**

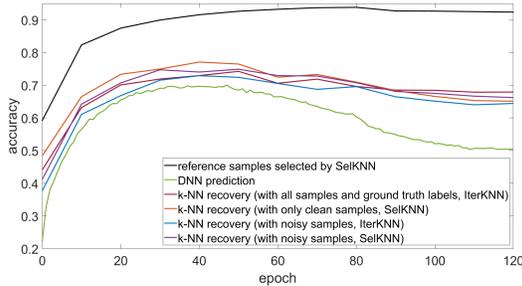Table 2: Label recovery rate(%) on CIFAR-10 with different symmetric noise levels. We run 10 trials, report mean and std.

Figure 3: Ground truth label inference accuracy in the first episode with different prediction methods, including DNN prediction, IterKNN prediction using noisy labels or ground-truth labels, SelKNN prediction using filtered labels or known clean labels (20% of whole dataset). The top curve is the clean rate of reference samples, which converges to around 92%. (CIFAR-10 with 60% symmetric noisy labels)

4.4 Iterative Retraining

Iterative retraining can correct more noisy labels and further improve the performance of the classifier, especially when noise level is extremely high. As shown in Fig. 4, the main proportion of label recovery is achieved in the first three episodes, yet some incremental recovery happens in the remaining episodes. The main purpose of retraining from scratch is to escape from over-fitting to the current noisy labels. Another approach to overcome over-fitting is to continue training but periodically reschedule the learning rate. When labels have been updated, the learning rate is also reset to a large value so that the model can transit to an under-fitting state. As shown in Fig. 4, performance cannot be further improved for later episodes if training continues without resetting the learning rate. It is because with small learning rate the model cannot escape from over-fitting noisy labels. We also found out retraining from re-initialized neural network performs slight better than simply rescheduling learning rate since
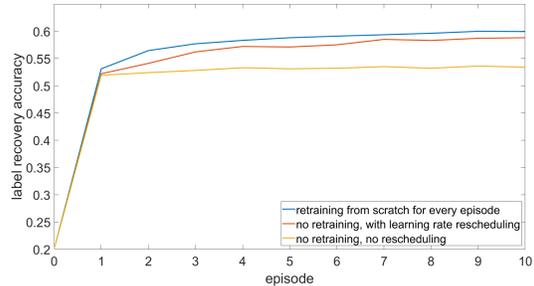

Figure 4: Label recovery rate for SelKNN under different training settings versus number of episodes. "No retraining" means training continues on the current neural network instead of re-initialized from scratch for every next episode. (CIFAR-10 with 80% symmetric noisy labels)

re-initialization makes neural network completely forget noisy labels. We compare the final recovery accuracy with baselines in Table 2. IterKNN performs well when noise level is low while SelKNN beats other methods on highly noisy dataset.
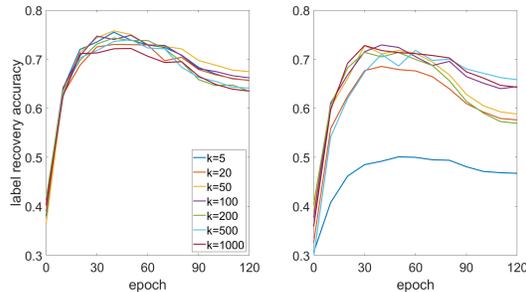

Figure 5: Ground-truth label inference accuracy of SelKNN (left) and IterKNN (right) with different k in the first episode. (CIFAR-10 with 60% symmetric noisy labels)

4.5 Impact of k Value

Since k value can have a large impact on the performance of neighbor classification, we also investigate how IterKNN and SelKNN can be affected by k. We select 7 different k values $(5, 20, 50, 100, 200, 500, 1000)$ and track the label prediction accuracy within one training episode without any label correction. The prediction is made after every 10 epoch and is shown in Fig. 5. As can be seen, SelKNN is relatively invariant to k and all the

9

choices of k consistently outperform the label prediction from classifier itself during the episode. It is also clear that IterKNN is more sensitive to k and requires a reasonably large k value to obtain good performances. This is because for IterKNN, the whole noisy training dataset is used as reference for KNN prediction. When k is too small, it is more likely all nearest neighbors are corrupted. So even if all the neighbors as well as the queried image belong to the same ground-truth class, KNN will still infer an incorrect labels.
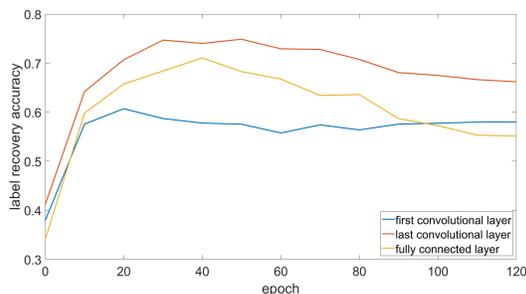

Figure 6: Ground-truth label inference accuracy of SelKNN with different layer features in the first episode. (CIFAR-10 with 60% symmetric noisy labels)

4.6 Choice of Deep Features

A fundamental question for KNN-based approaches is what feature to use. In order to evaluate the quality of different features, we train Pre-act ResNet 32 for one episode on CIFAR-10. Then we obtain t-SNE 2D embeddings from the outputs of three different layers as shown in Fig. 2.

As can be seen, features extracted from shallow layers are mixed together regardless of their classes. Conversely, features extracted from the last convolutional layer and the fully connected layer are clustered together if they belong to a same class, and separated from each other if they are from different classes. This observation indicates deeper features are more useful than shallow features for KNN, which is consistent with the curve shown in Fig. 6. Meanwhile, the recovery rate of using the last convolution layer as embeddings outperforms the others. We believe the fully connected layer is not suitable for label recovery either, because it is directly related to the logit layer, which is highly likely to be corrupted during training on

noisy dataset.

methods	CE	Joint Opt	Pencil	SelKNN
accuracy (%)	68.8	72.23	73.49	**76.78**

Table 3: Compare proposed method with baselines on Clothing1M.

4.7 Experiments on Real-world Noisy Dataset

Finally, we test our approach on Clothing1M database [32], a large real-world dataset composed of clothing images crawled from online shopping websites. Clothing1M comprises 1 million images with real noisy labels with additional 48 thousands verified clean data for training. Its overall noise proportion is approximately 38%. We adopt Resnet-50 pretrained on ImageNet as backbone. The data preprocessing procedure includes resizing the image with a short edge of 256 and randomly cropping a 224×224 patch from the resized image. We use the Adam optimizer and the weight decay factor is 0.005. The initial learning rate is 0.002 and decreased by 10 every 5 epochs. The batch size is 64.

Since the dataset is very large, it is not practical to search KNN across all the data samples. Therefore we only apply SelKNN for label correction, where the top 10,000 images are selected from each class as reference. There are 14 classes, thus in every round of label correction, our KNN database has 140,000 images in total. We set $k = 500$ and use L2 distance metric with majority voting for classification. We perform in total 5 training episodes and label updates, where each episode contains 15 epochs. The comparison with other 4 approaches is shown in Table 3. Our method achieves state-of-the-art accuracy and is more than 3% higher than the second best method.

5 Conclusion

In this paper, we propose k-nearest neighbor based iterative label correction framework for learning on the corrupted dataset. The approach is effective due to the robustness nature of KNN. We apply iterative retraining after every round of label correction to escape from over-

fitting. To further mitigate the impact of wrong labels, we use loss ranking to select clean samples as reference for KNN classification. We have conducted abundant experiments on both synthetic and real-world datasets. Empirical results show that our SelKNN algorithm achieves state-of-the-art performance. An interesting related direction is to study how to leverage deep KNN to defend backdoor attack [6].

References

[1] Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al.: A closer look at memorization in deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 233–242. JMLR. org (2017)

[2] Bahri, D., Jiang, H., Gupta, M.: Deep k-nn for noisy labels (2020)

[3] Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 41–48. ICML '09, ACM (2009)

[4] Braun, S., Neil, D., Liu, S.: A curriculum learning method for improved noise robustness in automatic speech recognition. In: 2017 25th European Signal Processing Conference (EUSIPCO). pp. 548–552 (2017)

[5] Brodley, C.E., Friedl, M.A.: Identifying mislabeled training data. Journal of artificial intelligence research **11**, 131–167 (1999)

[6] Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering (2018)

[7] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection (2005)

[8] Gao, W., Yang, B.B., Zhou, Z.H.: On the resistance of nearest neighbor to random noisy labels. arXiv preprint arXiv:1607.07526 (2016)

[9] Goldberger, J., Ben-Reuven, E.: Training deep neural-networks using a noise adaptation layer (2016)

[10] Hacohen, G., Weinshall, D.: On the power of curriculum learning in training deep networks. CoRR **abs/1904.03626** (2019), http://arxiv.org/abs/1904.03626

[11] He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision (2016)

[12] Huang, J., Qu, L., Jia, R., Zhao, B.: O2u-net: A simple noisy label detection approach for deep neural networks. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)

[13] Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Label propagation for deep semi-supervised learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

[14] Itqon, S.K., Satoru, I.: Improving performance of k-nearest neighbor classifier by test features. Springer Transactions of the Institute of Electronics, Information and Communication Engineers (2001)

[15] Jindal, I., Nokleby, M., Chen, X.: Learning deep networks from noisy labels with dropout regularization. 2016 IEEE 16th International Conference on Data Mining (ICDM) (Dec 2016). https://doi.org/10.1109/icdm.2016.0121, http://dx.doi.org/10.1109/ICDM.2016.0121

[16] Li, M., Soltanolkotabi, M., Oymak, S.: Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. CoRR **abs/1903.11680** (2019), http://arxiv.org/abs/1903.11680

[17] Liu, T., Tao, D.: Classification with noisy labels by importance reweighting. IEEE Transactions on pattern analysis and machine intelligence **38**(3), 447–461 (2015)

[18] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)

[19] Menon, A., Van Rooyen, B., Ong, C.S., Williamson, B.: Learning from corrupted binary labels via class-probability estimation. In: International Conference on Machine Learning. pp. 125–134 (2015)

[20] Papernot, N., McDaniel, P.: Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning (2018)

[21] Parvin, H., Alizadeh, H., Minati, B.: A modification on k-nearest neighbor classifier. Global Journal of Computer Science and Technology (2010)

[22] Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., Qu, L.: Making deep neural networks robust to label noise: A loss correction approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1944–1952 (2017)

[23] Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., Rabinovich, A.: Training deep neural networks on noisy labels with bootstrapping (2014)

[24] Sajjadi, M., Javanmardi, M., Tasdizen, T.: Mutual exclusivity loss for semi-supervised deep learning. 2016 IEEE International Conference on Image Processing (ICIP) (Sep 2016). https://doi.org/10.1109/icip.2016.7532690, http://dx.doi.org/10.1109/ICIP.2016.7532690

[25] Schroff, F., Criminisi, A., Zisserman, A.: Harvesting image databases from the web. IEEE transactions on pattern analysis and machine intelligence **33**(4), 754–766 (2010)

[26] Sitawarin, C., Wagner, D.: On the robustness of deep k-nearest neighbors. 2019 IEEE Security and Privacy Workshops (SPW) (May 2019). https://doi.org/10.1109/spw.2019.00014, http://dx.doi.org/10.1109/SPW.2019.00014

[27] Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., Fergus, R.: Training convolutional networks with noisy labels (2014)

[28] Tanaka, D., Ikami, D., Yamasaki, T., Aizawa, K.: Joint optimization framework for learning with noisy labels. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)

[29] Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results (2017)

[30] Wang, L., Liu, X., Yi, J., Zhou, Z.H., Hsieh, C.J.: Evaluating the robustness of nearest neighbor classifiers: A primal-dual perspective (2019)

[31] Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., Bailey, J.: Symmetric cross entropy for robust learning with noisy labels. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)

[32] Xiao, T., Xia, T., Yang, Y., Huang, C., Wang, X.: Learning from massive noisy labeled data for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2691–2699 (2015)

[33] Yan, Y., Rosales, R., Fung, G., Subramanian, R., Dy, J.: Learning from multiple annotators with varying expertise. Machine learning **95**(3), 291–327 (2014)

[34] Yang, J., Sun, X., Lai, Y.K., Zheng, L., Cheng, M.M.: Recognition from web data: A progressive filtering approach. IEEE Transactions on Image Processing **27**(11), 5303–5315 (2018)

[35] Yi, K., Wu, J.: Probabilistic end-to-end noise correction for learning with noisy labels. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

[36] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530 (2016)