2018 IEEE Winter Conference on Applications of Computer Vision

Iterative Cross Learning on Noisy Labels

Bodi Yuan[1], Jianyu Chen[1], Weidong Zhang[2], Hung-Shuo Tai[2], and Sara McMains[1]

[1]UC Berkeley
[2]Big Data Division, JD.COM American Technologies Corporation

Abstract

To address the problem of incorrect labels in training data for deep learning, we propose a novel and simple training strategy, Iterative Cross Learning (ICL), that significantly improves the classification accuracy of neural networks with training data that has noisy labels. We randomly partition the noisy training data into multiple separate subsets, each of which is used to train an independent network. After these independent networks predict labels for the original data, if the labels agree, we update the label with the predicted result for that data point, but otherwise we update the label with a random label, a key to the success of our proposed method. The process is repeated, possibly with several stages, to gradually improve the performance. Testing our method on MNIST and CIFAR-10 with partially shuffled labels, ICL significantly improves the classification accuracy of existing methods when the data labels have noise, especially in heavy noise situations. Moreover, the proposed method doesn't require any change to the underlying neural networks' structure or loss function, so it can also easily be combined with other existing methods that address noisy labels, improving their performance.

1. Introduction

Deep learning has revolutionized AI, including delivering the current state-of-the-art in visual object recognition [11]. One of the keys to deep learning is well labeled data [3, 16], with correct labels, without noise. However, in many situations, it is hard to get a completely clean data set, without extensive effort to manually clean up the data. So in practice, data often has noisy labels, from being manually mislabeled, or mislabeled by inaccurate but fast and cheap automated algorithms, or even because some portion of the data originally without labels is intentionally given random labels for supervised learning. Figure 1 shows an example of data sets corrupted with noisy labels.

Therefore supervised learning algorithms dealing with noisy labels are an active area of research. Sukhbaatar et al. [23] proposed a modification to a convolutional neural network that adds a noise layer to match the noise distribution and achieve the desired performance. Jindal et al. [5] proposed augmenting a standard network with a linear noise model layer at the end that learns the noise distribution, it can be removed after it has helped train the standard network to make accurate predictions in the presence of noise. Modifying the loss function with bootstrapping [21] gives more credence to predicted labels that are the same as the original labels. Natarajan et al. [17] addressed mislabeled data for the binary classification problem, using a simple unbiased estimator and a weighted surrogate loss to reduce the effect of noisy data. Xiao et al. [25] proposed a general training framework that models the relationship among images, labels, and noise with a probabilistic model, first predicting the type of noise and then removing the noise. An extra noise layer [1] is able to simultaneously learn both the neural network parameters and the noise distribution by assuming that the observed labels were created from the true labels by passing through a noisy channel whose parameters are unknown.

In semi-supervised learning, if a portion of the input is well-labeled and the remainder is originally unlabeled, noisy labels may come from assigning random labels to the unlabeled data, as described above. Lee [12] proposed a simple method to generate pseudo-labels for semi-supervised learning. It picks the class that has the maximal predicted probability as the label for each unlabeled data point and then trains networks in a supervised fashion. Kingma et al. [7] proposed a deep generative model to improve semi-supervised learning performance by generative approaches. By modifying the cost function, Rasmus et al. [20] proposed a ladder network that is trained to simultaneously minimize the sum of supervised and unsupervised cost functions by backpropagation. Co-training

[1]{bodiyuan, jianyuchen, mcmains}@berkeley.edu, Berkeley, CA
[2]{weidong.zhang, david.tai}@jd.com, Santa Clara, CA

978-1-5386-4886-5/18/$31.00 ©2018 IEEE
DOI 10.1109/WACV.2018.00088

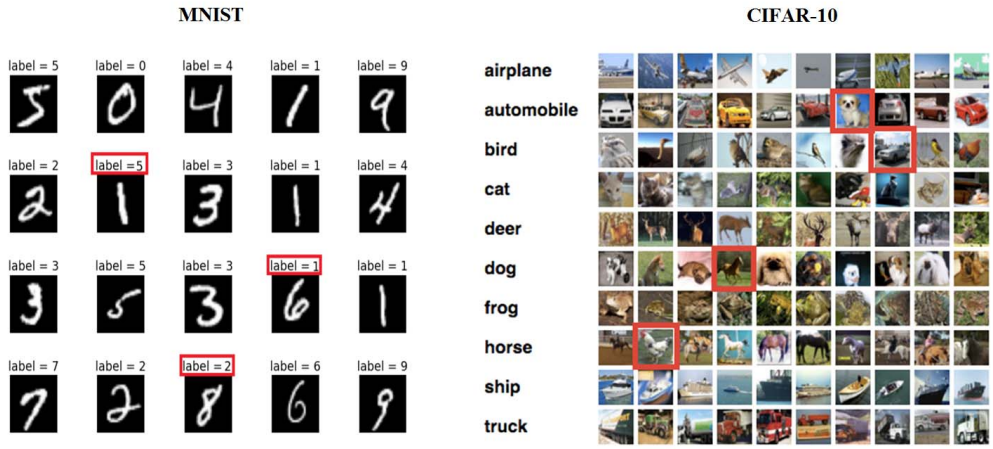


Figure 1: *Examples of training data sets corrupted with noisy labels*

[2, 18, 24, 13, 9, 14] for semi-supervised learning requires different views of a dataset, in other words, different features describing the data. It would be ideal for co-training to have two conditionally independent feature sets describing the same data. However, we don't always have that ideal situation, since the two feature sets might not be conditionally independent from each other and multiple feature sets may not be available. Similar to co-training, ICL also makes use of different features, but doesn't require different feature sets. Our proposed method uses a convolutional neural network that learns features [26] by itself rather than manually extracting features like in co-training. We feed our networks different datasets, instead of different features of the same dataset, to make it possible for the different networks to automatically learn some different features. Moreover, co-training's classifier will only gradually generate new labels for data points; it never doubts the labels it has generated, potentially introducing a large bias into the dataset when it outputs incorrect results. Furthermore it cannot correct the labels it has misclassified and cannot address incorrect labels. ICL uses multiple neural networks that learn independently and cross predict labels. It doesn't assume any label is a correct label a priori and it repeatedly updates all the labels in the whole dataset. Thus the features will also be updated after each stage, so ICL tends to have less bias than co-training because the networks are able to correct each other's classifications.

Another hot topic in the deep learning area, generative adversarial nets (GANs) [4, 15, 19, 22], also inspired ICL. The key idea of GANs is to use two separate competitive models, the generator and the discriminator. The competitors compete with each other, gradually learning a better model. In ICL, we also use two separate models, learner 1 and learner 2, but instead of competing, they learn from different content. Then they help each other verify their learning results and identify learning differences to enable more accurate re-learning. Learner 1 and learner 2 in ICL are parallel relations, whereas the generator and the discriminator in GAN are serial relations.

In this paper, we propose an effective and simple approach, Iterative Cross Learning, to train a neural network, which can significantly reduce the effect of label noise, maintaining the classifier's performance close to the performance trained with clean data. ICL is a general training strategy. It doesn't require changing the network structure or the loss function, which makes it possible to combine with and improve the performance of any type of neural network, including those that also use other techniques that address noisy labels.

2. Noisy Labels

For a k-classification problem, we will denote a training data point (x_l, y_l), where y_l is the correct label for x_l, $(y_l \in 1, 2, ..., k)$, where k is the total number of classes. However, in practice, some of the training data points might accidentally be (x_l, y_l^*), where the label y_l^* also belongs to $1, 2, ..., k$ but $y_l^* \neq y_l$. For example, in the CIFAR-10 dataset, a dog image might be mistakenly labeled as "cat" or "frog," as shown in Fig. 1. We call y_l^* a "noisy label."

In practice, noise may also be in the form of non-labeled data points. For the convenience of supervised learning, we can give a random label $1, 2, ..., k$ to each such non-labeled data point.

In this paper, we compare the performance of existing CNN models trained with and without using ICL on noisy labels. We use uniform noise for our experiments, because the repeatability of non-uniform noise experiments is low. To denote the amount of noise, we use "noise level" η for the fraction of labels "flipped." For a given data

point (x_l, y_l), there are two ways to flip its label. We can pick a random label from $1, 2, ..., k$ for y_l^* while ensuring that $y_l \neq y_l^*$, or we can just randomly pick a label from $1, 2, ..., k$ for y_l^*, which means there is a $1/k$ chance that $y_l = y_l^*$. We choose the latter for reporting our results to be consistent with [5]. Using confusion matrix M to represent the noise distribution, each element p_{ij} in M represents the probability of a member of class i being labeled as class j. So for a clean dataset, M will be an identity matrix. For the uniform noise model with noise level η, $p_{ii} = 1 - \eta + \eta/k$ and $p_{ij} = \eta/k$ $(i \neq j)$.

Since ICL is based on the assumption that the neural network can learn useful information even with noisy labels, let us consider their effect. For a k-classification problem with uniform noise level η, each class i will have expected fraction $(1 - \eta + \eta/k)$ correct labels, and expected fraction $(\eta - \eta/k) = (\eta/k) * (k-1)$ labels incorrectly set, distributed equally between labels $1, 2, ..., i-1, i+1, ..., k$. Let us consider the relationship between the noise in two classes i and j $(i \neq j)$. There will be fraction $1 - \eta + \eta/k$ of class i correctly labeled, with fraction η/k incorrectly labeled as j. Symmetrically, class j has fraction $1 - \eta + \eta/k$ correct labels and η/k incorrectly labeled as i. Between these two classes, the proportion P of correct to incorrect labels is

$$P = \frac{1 - \eta + \eta/k}{\eta/k} = \frac{1 - \eta}{\eta/k} + 1 \quad .$$

If there is some noise but not 100% noise, in other words, $0 < \eta < 1$, P will always be greater than 1, which suggests that the correct information will dominate between any two classes. So if we give the neural network unlimited time to learn and there is not overfitting, the network should finally tend to learn some correct information. If $\eta = 1$, then $P = 1$; all the labels are totally random, so the labels do not give any useful information and the network should learn nothing from the data. In other words, data with 100% noise is no better than unlabeled data, a problem for unsupervised learning, which is not the focus of this paper. We report results with noise levels as high as $\eta = 0.7$.

3. Iterative Cross Learning (ICL)

Sometimes even if the labels are noisy, existing CNN models can achieve an error rate lower than the noise level (this is confirmed in our experimental results). Thus, we can use the trained model to re-predict labels to make the labels less noisy. Based on those new labels we predict, if we are still able to train a model with an error rate lower than the current labels' noise level, then we can repeat this process to make the data less and less noisy.

As summarized in Figure 2, the key idea of Iterative Cross Learning is to train independent convolutional neural networks from different data, enabling these independent networks to clean up each other's data for the next learning stage. In this paper, for simplicity we will describe the algorithm for just two independent networks. For a given dataset S with noisy labels, we shuffle and partition the noisy data into two separate training datasets S_1 and S_2 with the same number of data points. Though these two datasets will tend to have similar noise levels and similar noise distributions, the data points in the two datasets will be different.

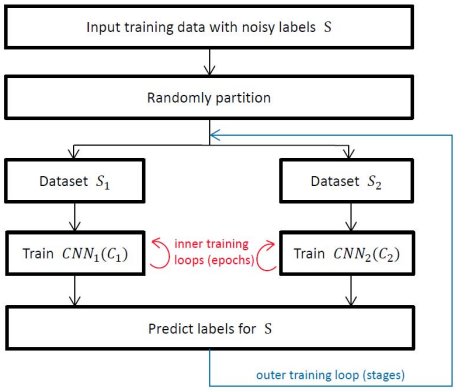


Figure 2: *ICL flow diagram*

Two convolutional networks C_1 and C_2 (which need not have the same structure, though for the currently reported results we use the same structure for convenience) will be trained on datasets S_1 and S_2 independently (the inner training loops in Fig. 2). Even with the same initial data and the same weight initialization, trained networks will not end up with the same weights if the input order is shuffled. Since the two networks will have different randomly initialized weights and are trained on different data, the parameters they learn will definitely be different from each other. The classifier C_1 might be a little better at classifying class i and the classifier C_2 might be a little better at classifying class j. ICL exploits this difference (the outer training loop in Fig. 2).

When we do the training with a noisy dataset, the performance might decrease after some inner training loop epoch because the network overfits noisy labels. So it is better to monitor the performance during the training and to decide which epoch's model to be used based on the monitoring performance. In some situations, besides the noisy training dataset S, we may have a clean set S_m without noisy labels that can be used to monitor training. For example, in practice, we may expend some human effort to manually clean up a small portion of the data. Since the training labels cannot be trusted, the "training accuracy" and "training loss" calculated from these noisy labels won't be particularly useful. Therefore, during training, we use the monitoring set, which is more trustworthy than the training set, to monitor the performance of the two networks C_1 and C_2. For each

network, the model with the best performance on monitoring set S_m during the training process will be used as the training result to predict labels.

Algorithm 1 Iterative Cross Learning with monitoring set

$S \leftarrow$ training set with noisy labels
$S_m \leftarrow$ monitoring set with clean labels
$S_1, S_2 \leftarrow$ randomly partition S into two separated datasets
initialize weights of CNNs C_1 and C_2
initialize Accuracies Acc_{c1}, Acc_{c2}, Acc and Acc' to 0
repeat
  initialize weights of CNNs C_1' and C_2'
  $C_1' \leftarrow C_1'$ trained on S_1 with monitoring on S_m
  $C_2' \leftarrow C_2'$ trained on S_2 with monitoring on S_m
  $Acc_{c1} \leftarrow C_1'$'s accuracy on S_m
  $Acc_{c2} \leftarrow C_2'$'s accuracy on S_m
  $Acc' \leftarrow$ max(Acc_{c1}, Acc_{c2})
  if $Acc' <= Acc$ **then**
    return the more accurate of C_1 and C_2
  end if
  $C_1 \leftarrow C_1'$
  $C_2 \leftarrow C_2'$
  $Acc \leftarrow Acc'$
  $L_{C_1} \leftarrow$ labels predicted for S by C_1
  $L_{C_2} \leftarrow$ labels predicted for S by C_2
  for data point x in S **do**
    if x's label l is the same in L_{C_1} and L_{C_2} **then**
      set x's label to l
    else
      set x's label to a random label $l \in \{1, 2, ..., k\}$
    end if
  end for
until the max stage number // e.g. 10
return the more accurate of C_1 and C_2

However, clean data is not always available or easy to obtain. If we don't have any clean data, we can still use our strategy without monitoring on a monitoring set. Training will just be terminated at a given training epoch number. Then we will use the trained model to predict labels for the dataset. Compared to monitoring on clean data, this will give slightly worse results, since we are not able to determine if a model from an earlier epoch might have been the model with the best performance on the clean dataset. (Algorithm 1 and Algorithm 2 provide the pseudo code for ICL with and without a monitoring set, respectively.)

After training, we use the two networks independently to predict labels for the original training dataset S. If the labels predicted by the two networks are the same, we set the data's label to match the prediction. Otherwise, we update the label with a random label. This technique will make the noise from incorrect predictions more uniformly random and less structured, which can give the trained networks less bias. The two networks learn from their different input at first, but then they also learn from each other, verifying each other's learning results and pointing out learning disagreements to re-learn; hence the name "cross learning."

Algorithm 2 Iterative Cross Learning without monitoring set

$S \leftarrow$ training set with noisy labels
$S_1, S_2 \leftarrow$ randomly partition S into two separated datasets
repeat
  initialize weights of CNNs C_1 and C_2
  $C_1 \leftarrow C_1$ trained on S_1 with max epoch
  $C_2 \leftarrow C_2$ trained on S_2 with max epoch
  $L_{C_1} \leftarrow$ labels predicted for S by C_1
  $L_{C_2} \leftarrow$ labels predicted for S by C_2
  for data point x in S **do**
    if x's label l is the same in L_{C_1} and L_{C_2} **then**
      set x's label to l
    else
      set x's label to a random label $l \in \{1, 2, ..., k\}$
    end if
  end for
until the max stage number // zero-indexed;
                // we use 1 as max
return C_1 or C_2

With a clean monitoring set, we could repeat the whole process until the change in monitoring accuracy between successive stages falls below some user-defined threshold. However, based on our experimental results, we found that most improvements are made in the first several stages, even when it took many more stages for ultimate convergence. So in practice, we terminate the training process if the monitoring accuracy does not increase after a new stage. (Such a decrease in accuracy suggests that the noise level from the two classifiers' prediction is already close to the noise level of the training data.) Without a clean monitoring set, we just repeat the process with a given maximum number of stages. Based on our experiments, depending on how challenging the input is, without a monitoring set the actual error rate may actually get worse again subsequent to the first round of retraining. Therefore we just set the max stage number to 1 (one round of retraining after updating the noisy labels based on the round zero results). Every time we update a data point with a random label, it brings random noise into the system such that some labels that were previously incorrectly predicted may get a chance to flip and help the networks recover from errors caused by misleading labels. In other words, this random flipping process enables networks to rectify some previous incorrect predictions.

When we finish the entire training process, we need to

choose one model from C_1 and C_2. If we have a clean monitoring set S_m, we choose the one that has better performance on S_m. Otherwise, we just randomly pick one as the final model.

We found that it tends to be slightly better if we only partition the data once for all the stages rather than randomly re-partition in every stage. We speculate the reason could be that some data points may be easier for learning from and some may be harder. Thus the one-time random partition may make the two datasets S_1 and S_2 have slightly different difficulty for classification, which means one dataset could make it easier to train a better classifier from that dataset. Compared to this, repeating random partitioning will eventually on average have two classifiers learn from an average difficulty dataset. So the training result might tend to be a little bit worse than using one-time random partitioning.

Note: The best we can achieve is that we perfectly predict the labels for all the data points in the original training dataset and the performance should be the same as training with clean data. However, in practice, we may not perfectly predict labels, so performance training with noisy data should always be worse than training with clean data.

4. Experiments and Results

We test our method on two popular datasets (MNIST and CIFAR-10) using TensorFlow, with uniform noise added. To be consistent with how noise level is measured in the previous work to which we compare our results, we use the same methodology and terminology as Jindal et al. [5]. Since both of the two datasets have 10 different labels $0, 1, 2..., 9$, for a given "noise level" η, we randomly choose η portion of the data and set a random label from 0 to 9 for each. For each dataset, we do experiments with adding 30%, 50% and 70% "noise" to training labels, respectively. (Note that because there are 10 possible labels, 10% of the time the "noise" will just be the original, correct label.)

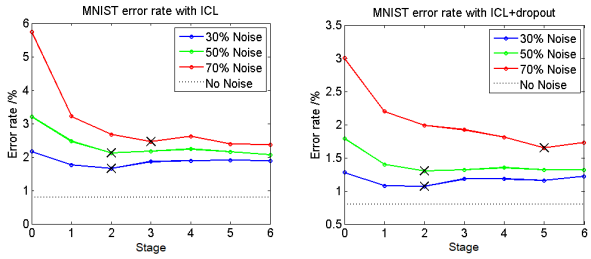
Using ICL, the original training labels will be updated in each stage. The training data will become cleaner and cleaner, which should decrease the error rate. So during the ICL process, the gap between the prediction error and noise level will become smaller and smaller. The improvement after the first several stages will also be small. And finally, after several stages, the error rate tends to be close to the noise level and the monitoring accuracy stops increasing. After the best performance stage, the accuracy starts to fluctuate. ICL's improvement after the first stage is always the greatest.

For each dataset, we compare the performance of the base CNN model, the true noise linear model in [23], and the dropout regularization noise model using [5] to ICL, both ICL combined with the base CNN model and ICL combined with the dropout regularization noise model.

4.1. MNIST with clean data for monitoring

The MNIST dataset is a set of handwritten digit images [10]. We use its 50,000 image training set, and randomly divide the other 10,000 images into 5,000 images for monitoring and 5,000 images for testing. Each image has a dimension of 28×28. We use a base CNN that has 2 conv-layers with 32 feature maps of 5×5 kernels and 64 feature maps of 5×5 kernels respectively, and each conv-layer is followed by ReLU activation and a 2×2 max-pooling layer. Next a dense layer of 1024 hidden units is fully connected with the second convolutional layer, followed by ReLU and dropout rate of 0.5, and fully connected with the final output layer of 10 units. We choose the Adam optimizer [6] and a learning rate of 1.0×10^{-4}.

Fig. 3 and Table 1 show our experimental results. Using ICL with the base CNN, the error rate decreases in the first several stages. Stage 0 is the baseline performance without using ICL. For 30% and 50% noise, the model achieves the best performance at stage 2. For 70% noise, the model achieves the best performance at stage 6. But in practice, we would terminate ICL when we find the performance gets worse at stage 4, returning the model from stage 3; its performance is already quite close to the best performance of stage 6. Combining ICL with the dropout regularization noise model [5], the best performance is achieved at stage 2 for 30% and 50% noise, and at stage 5 for 70% noise.



(a) ICL + base CNN (b) ICL + dropout

Figure 3: *ICL on MNIST with clean data for monitoring ("X" marks the stage returned by the algorithm).*

On the MNIST dataset, we find that the accuracy stops increasing significantly after no more than five stages (Fig. 3). (For comparison, we plot the results for a full six stages.) In practice, we stop as long as the accuracy stops increasing after a new stage, and return the prior model, which had the best performance.

We also visualize confusion matrices M after each training stage. We give one example in Fig. 4 of 70% noise to show how the noise distribution changes over training. Note: for clearer visualization, we omit all the diagonal matrix elements (marked NA); otherwise normalizing the confusion matrices will make the intensity of diagonal elements so strong that the other elements will all appear to be

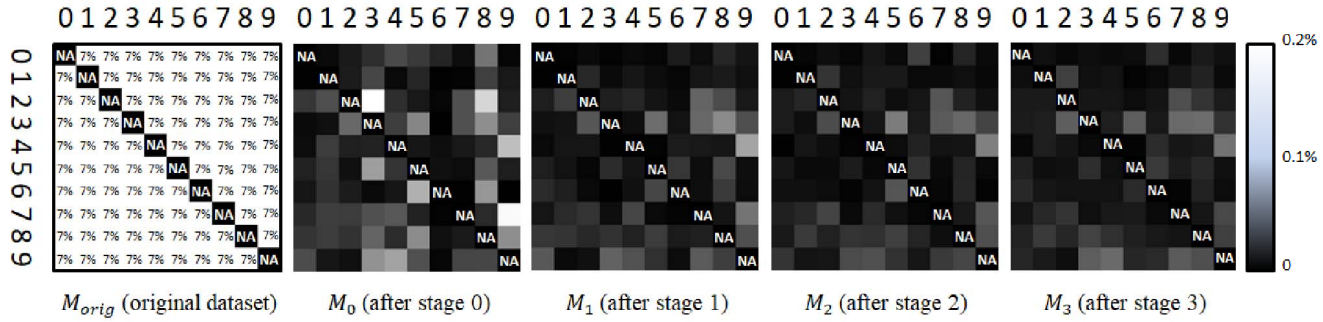


Figure 4: *Confusion matrices (base CNN with ICL training on 70% noise in MNIST dataset with clean data monitoring)*

equally dark.

From these visualization results we can see that the number of misclassified labels is less and less over training, which is also consistent with the decreasing error rate in the first several ICL training stages. In addition, we can see that the elements of the confusion matrix after stage 0 have the most contrast with each other, which means that labels predicted at that time also have the most structured noise. As seen in the visualizations of later stages in Fig. 4, our method continues to make improvements at these stages even with structured noise.

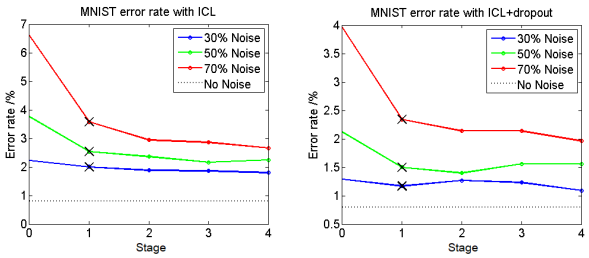
"Noise level"	30%	50%	70%
Base CNN	2.17	3.2	5.75
True noise[1]	1.3	2.06	3.31
Dropout[2]	1.25 (1.2)	1.8 (1.92)	3.01 (3.12)
Base CNN + ICL[3]	**1.66**	**2.12**	**2.54**
Dropout + ICL[3]	**1.07**	**1.32**	**1.78**

Table 1: *Error rates % for MNIST with clean data for monitoring; "noise level" is the percent of labels randomly reassigned.*

No matter which method it is combined with, ICL improves performance (see Table 1). Combined with the dropout regularization noise model, ICL achieves the best performance. The higher the noise level, the higher relative performance improvement we see with ICL.

4.2. MNIST without clean data

Sometimes clean data may not be available for monitoring, so we also test our algorithm without clean data. We used the same 50,000 images for training and 5,000 images for testing. The max epoch number was set to 20. Results are shown in Fig. 5 and Table 2.



(a) ICL + base CNN (b) ICL + dropout

Figure 5: *ICL on MNIST without clean data ("X" marks the stage returned by the algorithm; it is set to be 1 if we don't have clean data for monitoring).*

Unlike ICL with clean data, we don't have a monitoring set for monitoring each stage, so we just perform one stage of ICL (i.e. max stages set to 1). The performance is a little bit worse than ICL with clean data for monitoring, which is to be expected.

"Noise level"	30%	50%	70%
Base CNN	2.29	3.76	6.60
True noise[1]	1.3	2.06	3.31
Dropout[2]	1.29 (1.2)	2.12 (1.92)	3.96 (3.12)
Base CNN + ICL[3]	**2.00**	**2.54**	**3.58**
Dropout + ICL[3]	**1.17**	**1.50**	**2.34**

Table 2: *Error rates % for MNIST without clean data; "noise level" is the percent of labels randomly reassigned.*

4.3. CIFAR-10 with clean data for monitoring

The CIFAR-10 dataset is a set of tiny color images of dimension $32 \times 32 \times 3$ [8]. We use its 50,000 training images, and randomly divided the other 10,000 images into 5,000 images for monitoring and 5,000 images for testing.

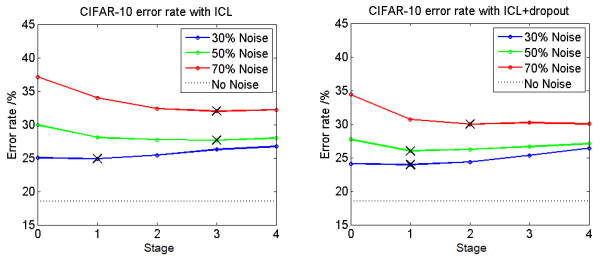
[1]Results using True Noise model as reported in Sukhbaatar et al. 2014 [23]. Since Jindal's model [5] already beats the true noise model, we don't re-implement the true noise model to combine with ICL.

[2]Model proposed by Jindal et al. 2016 [5]. Values out of parentheses are the results we measured using the methods in [5], and the values in parentheses are the results they reported in their paper.

[3]The numbers in bold are results from our algorithm.

Classical data augmentation techniques (random crop, random flip, random brightness and random contrast) are used for the training. We use a similar base CNN model as described above, except that the size of the max-pooling layer is 3×3 and the number of hidden units is 200.

Fig. 6 and Table 3 show experimental results. Using ICL alone on the base CNN, for a 30% noise level, ICL achieves the best performance at stage 1. For a 50% noise levels, it achieves the best performance at stage 3. And for a 70% noise level, it achieves the best performance at stage 4. Using ICL with the dropout regularization noise model [5], for 30% and 50% noise level, it achieves the best performance at stage 1. For 70% noise level, it achieves the best performance at stage 2.



(a) ICL + base CNN (b) ICL + dropout

Figure 6: *ICL on CIFAR-10 with clean data for monitoring ("X" marks the stage returned by the algorithm).*

"Noise level"	30%	50%	70%
Base CNN	25.1	29.9	37.1
True noise[1]	24.8	29.6	36.2
Dropout[2]	24.1 (24.4)	27.7 (32.6)	34.4 (33.0)
Base CNN + ICL[3]	**24.9**	**27.6**	**32.0**
Dropout + ICL[3]	**23.9**	**26.0**	**30.0**

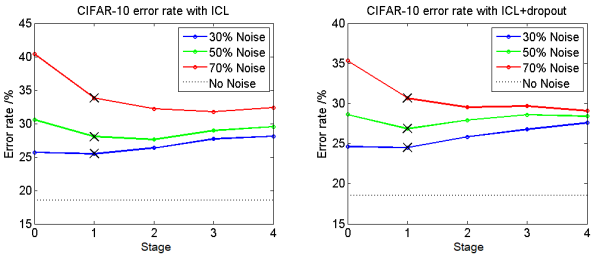
Table 3: *Error rates % for CIFAR-10 with clean data for monitoring; "noise level" is the percent of labels randomly reassigned.*

Again, using ICL improved the performance of both the base CNN and dropout regularization (see Table 3). However, unlike the results on MNIST, on CIFAR-10 the improvement at 30% noise is fairly small (the base CNN achieves a 25.1% error rate and ICL improves it to 24.9%; the dropout model achieves a 24.1% error rate and ICL improves it to 23.9%). The original base CNN, when trained on the uncorrupted 0% noise CIFAR-10 dataset, already has a 18.5% error rate, which is much higher than the corresponding 0.8% error rate on the uncorrupted MNIST dataset. When we train the model with 30% noise, it has around a 25% error rate, which is already very close to the 30% noise level. In this case, there is not much additional useful information that ICL can use for improvement with 30% noise.

Moreover, after the first stage of ICL, in such circumstances performance may only deteriorate in subsequent stages. Though we terminate the ICL process in practice if we find the performance does not improve after a new stage, for our experiments we ran ICL for four or more stages. We found that the performance with 30% noise actually would get worse and worse after the 1st stage on the CIFAR-10 dataset. If the error rate is close to the noise level, when we update the labels, we actually will bring more and more noise into the labels.

4.4. CIFAR-10 without clean data

We also tested our algorithm without clean data on the CIFAR-10 dataset, using the same 50,000 images for training and 5,000 images for testing (Fig. 7 and Table 4). The max epoch number was set to 100 and max stages to 1.



(a) ICL + base CNN (b) ICL + dropout

Figure 7: *ICL on CIFAR-10 without clean data ("X" marks the stage returned by the algorithm).*

"Noise level"	30%	50%	70%
Base CNN	25.7	30.6	40.4
True noise[1]	24.8	29.6	36.2
Dropout[2]	24.6 (24.4)	28.6 (32.6)	35.3 (33.0)
Base CNN + ICL[3]	**25.5**	**28.1**	**33.8**
Dropout + ICL[3]	**24.5**	**26.8**	**30.6**

Table 4: *Error rates % for CIFAR-10 without clean data; "noise level" is the percent of labels randomly reassigned.*

5. Discussion

Training on the uncorrupted MNIST training set, the base model can achieve a 0.8% error rate. With 30% noise, ICL reduces the dropout regularization noise model error from 1.25% to 1.07%, substantially reducing the distance from the ideal performance achievable when trained without noise (0.8%). To compare ICL's performance with different noise levels, the absolute improvement is not as intuitive a measure as the improvement relative to the ideal of training without noise. Therefore we also measure the

relative performance improvement. Define I_a, the absolute performance improvement, and I_a^*, the ideal absolute performance improvement, as

$$I_a = e - e_{ICL},\ I_a^* = e - e^*$$

where e is the model's error rate trained under noisy data, e_{ICL} is the model's error rate with using ICL, and e^* is the model's error rate trained under the original uncorrupted data. Then we define I_r, the relative performance improvement, as

$$I_r = I_a / I_a^* \ .$$

With this notation, the ideal performance is $e_{ICL} = e^*$, such that $I_r = 100\%$.

"Noise level"	30%	50%	70%
MNIST with clean data, ICL improvement to:			
Base CNN	37.2%	45.0%	64.8%
Dropout	40.0%	48.0%	55.7%
MNIST without clean data, ICL improvement to:			
Base CNN	19.5%	41.2%	52.1%
Dropout	24.5%	47.0%	51.3%
CIFAR-10 with clean data, ICL improvement to:			
Base CNN	3.0%	20.2%	27.4%
Dropout	3.6%	18.5%	27.7%
CIFAR-10 without clean data, ICL improvement to:			
Base CNN	2.8%	20.1%	30.1%
Dropout	1.6%	17.8%	28.0%

Table 5: *ICL relative improvement* I_r

The relative improvement numbers (see Table 5) illustrate how ICL is particularly advantageous under higher noise levels. Furthermore, looking at ICL's relative improvement under the same noise level and the same dataset, the relative improvements are very similar from combining ICL with these two different models, base CNN and dropout regularization. This also suggests that no matter which method it is combined with, ICL will hopefully improve performance.

The underlying model still plays an important role. The final performance with ICL highly depends on the underlying model's performance. The better performance the underlying model can achieve, the better final result ICL can achieve.

6. Limitations

ICL has a limitation that the underlying model, trained on uncorrupted data without ICL, must have a classification error rate lower than the noise level. If its error rate is higher than the noise level, the data will almost certainly be noisier after ICL's random flipping of labels for inputs on which the separate networks disagreed.

Another limitation of ICL is that it will take longer for training than training just the underlying networks it is combined with, since it needs to repeat the training for at least one stage. However, when we modify the networks' structure or tune the training hyper-parameters, we don't need to use ICL. We will only use ICL once the structure and hyper-parameters have already been chosen and tuned using standard training. So even if ICL takes multiple stages to train, it is only at the last step of training.

7. Conclusions

The experimental results show that our training strategy is able to greatly improve training performance when data labels are noisy.

Moreover, a great advantage of ICL is that it has virtually no parameters to choose, nor does it need to re-tune the hyper-parameters of the underlying model it is combined with. As long as the underlying model works well, ICL can be used to further improve its performance under noisy data. Furthermore, ICL could be combined with other existing or yet-to-be-invented models that address noisy labels, improving those models' performance without any cost except the training time. ICL is simple and easy to implement.

Numerous variations of the basic ICL method are envisioned. ICL doesn't rely on the structure of neural networks, so it can be used to train differently structured networks simultaneously, on different training subsets, rather than the same network structures. This could lead to different networks learning distinct features of the data, which might benefit the cross labeling process. In this paper, we performed the experiments with convolutional neural networks as the underlying model because CNNs' performance is usually better than other approaches on image classification tasks. However, ICL isn't restricted to CNNs; it could also be combined with other types of neural networks and even other machine learning algorithms. Moreover, our training strategy isn't limited to partitioning datasets into just two separate piece with two classifiers, though fewer classifiers means that each classifier can access more training data and the label updating rules can be simpler. Different numbers of partitions and different partitioning methods are another rich area of exploration for the ICL approach.

8. Acknowledgments

This work is supported by a UC Berkeley Graduate Fellowship and the company JD.COM. Thanks to helpful discussion with Alexei A. Efros, and feedback from Yan Kang, Kai Li, and the anonymous reviewers.

References

[1] A. J. Bekker and J. Goldberger. Training deep neural-networks based on unreliable labels. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2682–2686. IEEE, 2016.

[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on computational learning theory*, pages 92–100. ACM, 1998.

[3] X.-W. Chen and X. Lin. Big data deep learning: challenges and perspectives. *IEEE Access*, 2:514–525, 2014.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] I. Jindal, M. Nokleby, and X. Chen. Learning deep networks from noisy labels with dropout regularization. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 967–972. IEEE, 2016.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[8] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[9] A. Kumar and H. Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 393–400, 2011.

[10] Y. LeCun. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[11] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[12] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.

[13] A. Levin, P. A. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *ICCV*, pages 626–633, 2003.

[14] R. Mihalcea. Co-training and self-training for word sense disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, 2004.

[15] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[16] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, 2015.

[17] N. Natarajan, A. Tewari, I. S. Dhillon, and P. Ravikumar. Learning with noisy labels. In *Neural Information Processing Systems (NIPS)*, dec 2013.

[18] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on information and knowledge management*, pages 86–93. ACM, 2000.

[19] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[20] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

[21] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[22] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

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

[24] X. Wan. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-volume 1*, pages 235–243. Association for Computational Linguistics, 2009.

[25] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2699, 2015.

[26] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.