Addressing Noisy Label Problem: Iterative Cross Majority Learning with Convolutional Neural Network

Honghao Qiu
Stanford University
450 Serra Mall, Stanford, CA
honq@stanford.edu

Abstract

Data collected in real world setting are often corrupted with incorrect labels, such noise in training data can lead to significant model performance degrade on test set. To address noisy label problem, we propose a new simple yet effective method, Iterative Cross Majority Learning (ICML), to train multiple convolutional neural networks independently on noisy data, and update data label based on majority vote across predicted outputs from the trained models. We then repeat the process in multiple stages iteratively, and use the last stage trained CNNs to perform ensemble learning for making test set prediction. Our experiments on MNIST dataset shows that the proposed ICML method is able to achieve near state of the art result of over 97% prediction accuracy on test set after trained on data with noise level as high as 70%, and recovering over 90% of the noisy labels correctly on MNIST. Moreover, by introducing random labeling for disagreed predictions at early stages (i.e. Iterative Cross Learning - Random, or ICL-R), and applying ICML in later training stages, we find such combination provides even better result and converge faster than previous approaches. We further analyzed metrics and visualized network outputs to show the effectiveness of this method. Besides the effectiveness of proposed approach in noisy label training, it can also be used for label generation given a small portion of true label data (i.e. create random labels for unlabeled data and perform ICML label update to recover true labels).

1. Introduction

This paper addresses noisy label training problem in supervised learning, specifically, we investigates into modeling convolutional neural network with noisy image dataset (i.e. image data corrupted with noisy labels). An important problem in computer vision is to train models that has good predictive power given noisy labeled images which is common in real world. When data labels contain noise, models trained on such dataset suffers different levels of performance degrade at test time, and since data collected in real world typically contains noise due to human label error, data collection/label system imperfection, etc., it is important that we can train models on noisy label data and still be able to make accurate prediction on new unseen data, this is especially critical for security/safety sensitive computer vision applications such as face recognition, autonomous driving, and medical treatments etc. We aims to address the noisy label training problem in machine learning by proposing new methods that can create learners able to be trained on corrupted dataset and still achieves good performance at test time.

In this paper, we propose several effective yet simple approachs for noisy label problem, to train convalutional neural networks that can significantly reduce noisy label effects, recover true labels from noisy data, and improve model accuracy such that performance is close to model trained with clean data with true labels. The key idea is to perform Iterative Cross Majority Learning (ICML), which iteratively train multiple CNNs on noisy labels and update training data labels based on majority vote when prediction disagrees. Our proposed method is simple and generalized as it only requires label update in training time, without designing specific loss function, gradient update, or network architecture for noisy label problem, hence it can be used as a base method to plug in any type of model or network architectures, as well as combining with other methods in this research area such as weighted loss function for noisy training, to further improve model performance on noisy labels. Through experiments, we find that ICML method (along with random label update when CNNs outputs all disagree) is able to achieve 97% test accuracy with noise level as high as 70% in training labels.

For the experiments in this project, data input during training are noisy labeled images from MNIST (digit images from 0-9) with different noise level introduced, while input at test time are images with true labels. The output

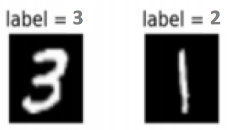


Figure 1. MNIST dataset: true label vs noisy label examples

of ICML model are predicted labels (digit classes from 0-9) for MNIST images. The goal is to be able to take corrupted label inputs at different noise level and be able to recover true labels from noisy dataset, and achieve accuracy close to model trained to clean data at test time.

Figure 1 shows two examples of MNIST data with true label and noisy (corrupted) label.

To further investigate into noisy label problem, We also analyzes what has caused the performance drop in noisy labeled data training, as well as visualizing CNN layer outputs and analyzing statistical differences between models trained with perfectly labeled data versus noisy labeled data.

2. Related Work

In recent years noisy label problem is gaining more and more attention, and multiple works have advanced this area of research aiming to address the problem of effective training neural network on noisy labeled data. These works can mostly be categorized into supervised learning, semi-supervised learning, and noise learning.

Supervised learning Extensive supervised learning research has been conducted on noisy label problem. Xiao et al. [24] proposed to model images, label noise relationship using probabilistic graphical model, and able to greatly correct noisy labels in the dataset with limited true labels provided for training. Some other works [19, 22] try to train learners that are noise agnostic with noisy labels provided and [1] modifies loss function to adjust for noise which gives more weights to prediction that consistent with true labels, but Bartlett et al. [3] proves that most loss functions are sensitive to noise in labels. On the other hand, similar to Xiao's work [24], some recent paper [5] achieves near state of art performance on noisy labels with more general training/modeling framework without twisting loss function or network architecture by directly learning on noise labels. Some papers also focus on cleaning up noise labels to correct ones or remove them once identified [6, 2], but these methods suffers from either over cleaning by removing too much low confidence noise data or hard to distinguish mislabeled ones from true labels.

Semi-supervised learning In semi-supervised learning, noisy labels are random/predicted labels assigned to unlabeled data, along with another portion of training data with true labels provided. Lee [14] proposed a semi-supervised learning approach to pick highest predicted probability class as label and then train model with these labels. Rasmus et al. [18] trained ladder network that to minimize aggregated supervised and unsupervised learning cost. On the other hand, co-training [4, 11, 15, 17] is another active semi-supervised learning research area that uses different and conditionally independent features for a given dataset. It is effective when features are I.I.D., but such assumption is strong and typically not valid for most real world data.

Noise modeling To model and learn noise pattern with deep neural network, Mnih and Hinton [16] proposed to model noise in aerial images with noise labels. Jindal et al. [9] and Sukhbaatar et al. [20] both proposed to train network with a noise layer to learn noise distribution, while Xiao et al. [21] proposed to model noise using probabilistic model. Another work from Larsen et al. [12] assumed independent noisy labels from true labels to address noisy label learning problem, which is a special case and less generalized for all datasets.

Our work falls into supervised learning by directly learning on data with noisy labels, it is generalized and does not make assumption on data or requires changes in loss function, so it is more flexible and able to apply to more scenarios. Our approach is also able to recover true labels from noisy data to great extent, and hence can be applied to semi-supervised learning use cases as well for automatic data label generation with a portion of true labels provided.

3. Methods

We propose following four methods to approach noisy labels problem:

- Iterative Cross Majority Learning (ICML): multiple CNN learners trained side by side and use majority vote of learners' predictions to perform label update
- Iterative Cross Learning - Random (ICL-R): multiple CNN learners trained side by side and performs random label update when labels disagree
- ICML + ICL-R Hybrid Label Update: when training three CNN learners, when all predicted labels are different, perform random label update, else performs majority vote
- ICML + ICL-R Multi-stage Learning: at early (0 to 1) training epoch/stages, use pure random label update strategy when learners disagrees, in later training epochs/stages, perform label update based on majority vote across CNN learners' predictions

3.1. Iterative Cross Majority Learning (ICML)

We first propose the following approach for addressing noisy label training - inspired by previous supervised learning researches in noisy labels, recent advances in Generative

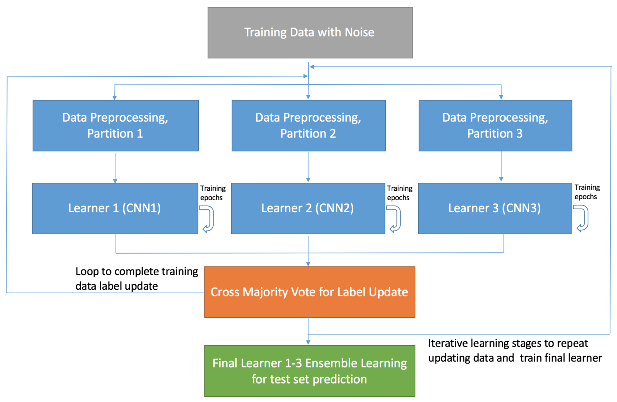

Training Data with Noise
Data Preprocessing, Partition 1
Data Preprocessing, Partition 2
Data Preprocessing, Partition 3
Learner 1 (CNN1)
Learner 2 (CNN2)
Learner 3 (CNN3)
Training epochs
Training epochs
Training epochs
Loop to complete training data label update
Cross Majority Vote for Label Update
Iterative learning stages to repeat updating data and train final learner
Final Learner 1-3 Ensemble Learning for test set prediction

Figure 2. Proposed Approach: Iterative Cross Majority Learning

Adversarial Network (GAN) [8] and majority vote in Ensemble Learning [7], we propose Iterative Cross Majority Learning (ICML) to train multiple (in this paper, 3) convolutional neural networks as independent learners on noisy data (data is randomly partitioned into multiple sets to train independent CNNs separately), if the predicted label agrees, the data label is updated with predicted label, if not, we use the majority voted label across CNN outputs to update data label. We would then repeat the process in multiple stages iteratively, reshuffle training data to make sure all labels are updated during this process, and use the last stage CNNs to perform ensemble learning for making prediction on test set.

Algorithm 1 Iterative Cross Majority Learning

```
 1: procedure
 2:     D ← training set with noise introduced
 3:     D' ← random partition, pre-processing
 4:     T ← test set with true labels
 5: repeat:
 6:     W ← initialize weights for 3 CNNs
 7:     $M_1, M_2, M_3$ ← CNNs trained with D' partitions
 8:     $L_1, L_2, L_3$ ← CNNs predicted labels
 9:     for x in D' do
10:     $L_x$ ← majority votes of CNNs predicted labels
11:     end for
12: until maximum training epochs reached
13: $M_1, M_2, M_3$ ensembles to predict on test set T
14: return test set prediction accuracy
```

Figure 2 shows a brief flow of the Iterative Cross Majority Learning approach.

We are testing the effectiveness of ICML method on image classification problem, and we choose CNN because the convolution filters are able to exploits features locality, identifies features like edges and circles, and are translational invariant along with pooling layer, therefore the learned CNNs are able to generalize its learnings well on unseen image dataset and make accurate predictions at test time given images with different poses/conditions. To design the 3 CNN networks used in our experiment, we follow a well-adopted architecture pattern in research papers: [Conv-ReLu-Maxpool]*N - Dense - ReLu - Dropout - Dense - Softmax prediction. Specifically, in order to create CNNs that learn different hypotheses on data, we make these 3 learners slightly different in architecture, assigning them with one, two and three [Conv-ReLu-Maxpool] modules respectively.

Figure 3 illustrates network architectures of the three CNN learners in all our experiments.

To make categorical predictions for each image data, we use softmax function as last layer of the model to compute prediction scores for each categories:

$$y_k = \frac{\exp(\phi_k)}{\sum_j^C \exp(\phi_j)}, \tag{1}$$

where k-th score is the score for k-th category, C is the number of total classes (e.g. 10 in MNIST case). And then we pick the class with highest softmax score as the prediction. To train these CNNs, we use multi-class cross entropy loss which extends cross entropy loss in binary classification case:

$$loss = -(ylog(p) + (1-y)log(1-p)) \tag{2}$$

by summing loss for each classes across all observations:

$$loss = -\sum_{j=1}^{C} y_{o,j} log(p_{o,j}) \tag{3}$$

where C is the number of total classes (e.g. 10 in MNIST case), y is a binary indicator for whether class label j is correct for observation o, and p is the predicted probability for observation o to be in class j. We feed data in mini-batches (100 data points per batch) for training, and we use ADAM optimizer to perform gradient update for CNN weights training.

Similarly, we use the same CNN network architectures for the 3 CNN leaners to be trained, as well as same multi-class cross entropy loss and ADAM optimizers designed for all the algorithms that follows in this section.

3.2. Iterative Cross Learning - Random (ICL-R)

Similar to ICML method, we can use Iterative Cross Learning - Random approach to update labels. The difference is that after training 3 CNNs to make predictions independently, instead of using majority vote strategy for label update, we use random label update (i.e. randomly choose label from 0-9 in MNIST case) to perform label update when CNNs prediction outcomes disagree. While

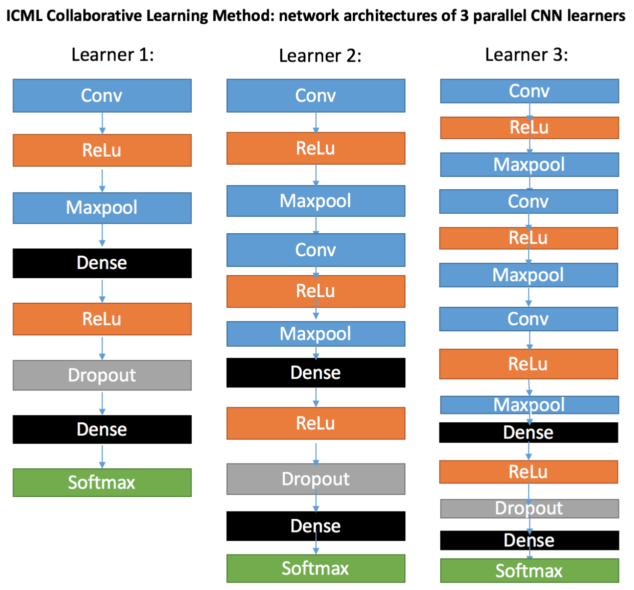


Figure 3. Network Architectures for parallel learners

ICML method is based on the assumption that learned features across models should lead to common label output if they are predicting correctly, we also need to consider the problem where labels might get stuck in a majority vote that is not actually a true label. So the intuition behind ICL-R method is to create randomness in labels when prediction disagrees, so that it won't be stuck in wrong output predictions, which reduces the effect of noisy labels to the prediction outcome, and overtime after multiple stages we expect the labels can gradually converge to the correct ones. A downside of ICL-R method compared to ICML method is that the time it takes for training to recover true labels and achieve comparable accuracy on test set is longer, as it takes more time to learn and converge due to randomness.

3.3. ICML + ICL-R: Hybrid Label Update

Taking advantages and disadvantages of majority udpate and random update into consideration, we further proposes a hybrid label update strategy, where after training multiple CNNs to make predictions on noisy labels data, we update the labels based on a hybrid approach: if the prediction outputs from three models are all different, implying the label is likely corrupted (majority vote from these are meaningless in this case as all three predicted labels are equally unlikely to be correct), we update the label by assigning it to a random class to reduce the effect of noisy label, and hope it can be re-updated to correct label in future stages. If there is common label prediction from at least a pair of CNNs, implying there is common features learned for prediction and it is more likely a correctly labeled data, we perform label majority update in this case. This method is expected to outperform pure ICML strategy as ICML might get stuck in wrong labels and not able to get out due to no randomness introduced to break tie.

3.4. ICML + ICL-R: Multi-stage Learning

Delving deeper into the training process of ICML and ICL-R, we find out that ICL-R is more effective at early stages, especially when noise level is high, since in this case the randomness we introduce offset the noisy label impact on performance degrade, and majority vote does not help much at the begining when noise level is very high (e.g. 70%) as all prediction outputs of 3 CNNs might be wrong. While in later stages, as CNNs learns more generalized feature patterns for the dataset, it is much more likely to make correct prediction from the 3 networks, and it makes more sense to perform majority vote update instead of random update since ICML at this stage is more likely to find true label and makes training much faster to converge. This Algorithm can be detailed as follows.

Algorithm 2 ICML + ICL-R: Multi-stage Learning

```
 1: procedure
 2:     $D \leftarrow$ training set with noise introduced
 3:     $D'$ ← *random partition, pre-processing*
 4:     $T \leftarrow$ test set with true labels
 5: repeat:
 6:     $W \leftarrow$ initialize weights for 3 CNNs
 7:     $M_1, M_2, M_3 \leftarrow$ CNNs trained with D' partitions
 8:     $L_1, L_2, L_3 \leftarrow$ CNNs predicted labels
 9:     for x in D' do
10:     if epoch $<$ early stage training threshold then
11:         $L_x \leftarrow$ assign random class labe
12:         Else
13:         $L_x \leftarrow$ majority votes of CNNs predicted labels
14:     end if
15:     end for
16: until maximum training epochs reached
17: $M_1, M_2, M_3$ ensembles to predict on test set T
18: return test set prediction accuracy
```

Figure 4 illustrates ICML + ICL-R multi-stage learning.

3.5. Baseline and Oracle Model

To evaluate the effectiveness of proposed approach, we need to compare it with a baseline model (lower bound estimate) and oracle model (upper bound estimate). Inputs and outputs for baseline/oracle models are the same as the method described above. In this paper, baseline model is a tuned CNN trained against corrupted noisy dataset directly with no label update approach applied, while Oracle model is a tuned CNN trained on original true label dataset. We would then use the accuracy and loss for baseline and oracle models to compare with proposed approach.

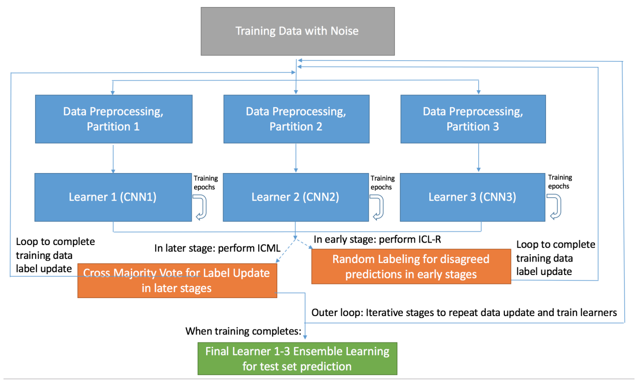

Training Data with Noise
Data Preprocessing, Partition 1
Data Preprocessing, Partition 2
Data Preprocessing, Partition 3
Learner 1 (CNN1)
Training epochs
Learner 2 (CNN2)
Training epochs
Learner 3 (CNN3)
Training epochs
Loop to complete training data label update
In later stage: perform ICML
In early stage: perform ICL-R
Loop to complete training data label update
Cross Majority Vote for Label Update in later stages
Random Labeling for disagreed predictions in early stages
Outer loop: Iterative stages to repeat data update and train learners
When training completes:
Final Learner 1-3 Ensemble Learning for test set prediction

Figure 4. ICML + ICL-R: Multi-stage Learning

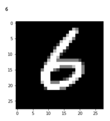


Figure 5. sample MNIST image: 28*28*1 resolution, grey-scaled

Furthermore, in this work we analyze the noisy label training problem by visualizing outputs at each CNN layers, computing training model statistics to compare differences and find the impact caused by noisy labels. Evaluations are both qualitatively (plots of figures for features learned) and quantitatively (report statistical metrics and model evaluation results such as accuracy, P/R and Confusion Matrix).

4. Dataset and Features

We use MNIST dataset [13] for experiments. There are 60,000 training images (resolution: 28*28*1 each), and 10,000 test images in total. We partition the training data into 55,000 training set and 5,000 validation set, and introduce noise to the labels at different levels: 30%, 50%, and 70% labels in training set are set to a different than true class label randomly. Note that we a set of 10,000 intact true labeled images aside for testing purpose.

We pre-process and normalize the image data to 0-1 value, subtracted mean of training images for each image data, then divide them by standard deviation of training images. In network architecture experiments, we also tested using batch normalization before non-linearity layer to force activations into gaussian distribution during training. Augmentation is also tried by performing flipping operations on MNIST images. Sample MNIST data shown in Figure 5.

We build and train CNNs to learn features directly from the data, features learned including edges, circles. As shown in Figure 6, where we visualize CNN output of the 1st conv layer to show features learned by the network, note that for each filters it has emphasis on slightly different local

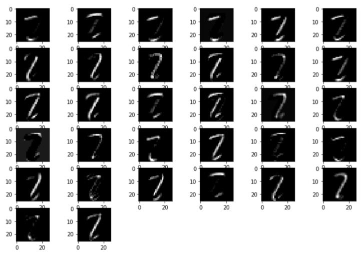


Figure 6. Visualize features learned by 1st convolutional layer

features from the dataset.

We then conduct combination of ICML and ICL-R training approaches on these MNIST data with noisy labels introduced at different levels, and compare the prediction result with models trained on original data.

5. Experiments/Results/Discussion

5.1. Experiments Setup

STARTER CODE: we used Stanford CS231N Simple Tensorflow 2-Layer CNN as starter code, and made adaptions in data partitioning, pre-processing, training process, evaluation functions, network architecture, etc., and most importantly, introduced majority voting, random labeling update methods as well as hybrid update strategy and multi-stage training in the experiments to evaluate effectiveness of our proposed approach. Github Code: https://github.com/honghaoq/Noise-CNN-Addressing-Noisy-Label-Problem-with-Convolutional-Neural-Network.

Model Hyper-Parameters/Network Setup: some hyper-parameters used in our experiments are - learning rate 2e-4, number of epochs 10, batch size 100, dropout ratio 0.2, noise level 0.3, 0.5, and 0.7, epoch threshold used to control ICML vs ICL-R update in multi-stage learning is 2, multi-class cross entropy loss for loss function, ADAM optimizer for gradient update, and network architecture as shown in Figure 3 in Method section. By tuning these parameters, we find that learning rate at 2e-4 performs well as it converges relatively fast in training and does not overshoot by performing too large update at each step, mini-batch 100 also balances well in training time and model effectiveness, and finally, we choose ADAM optimizer as it outperforms RMSProp or mini-batch stochastic gradient updates with its adaptive learning rate and robustness in training.

Quantitative & Qualitative Analysis: We use the following quantitative metrics in our experiments: 1) Accuracy: prediction accuracy, num of correctly predicted images/num of all images; 2) Error rate: prediction error rate on test set at different noise levels; 3) Loss: multi-class cross entropy loss computed during training; 4) Confusion matrix: matrix with each row representing instances in predicted

Noise level	Test Accuracy	Final Epoch Training Loss
30%	0.9791	2.0726
50%	0.9695	2.1451
70%	0.9596	2.2019

Table 1. ICML Method Accuracy & Loss trained on Noise Levels 30%, 50%, and 70%

class while each column representing instances in true class, which shows whether CNNs often mislabel one class as another; 5) Class visualization: visualized outputs for each conv layers in CNNs, etc. We also conduct qualitative reasoning and network visualizations for further analysis.

5.2. Iterative Cross Majority Learning (ICML)

We first experiment ICML training approach on noisy labels. For each run, we train three independent CNNs on noisy labels and update labels based on majority vote, the training is repeated 10 epochs, then we use the last stage CNNs to perform ensemble learning to predict on test set. By running training ten times and averaging the results, its accuracy and loss trained on different noise levels (30%, 50%, and 70%) are captured in Table 1.

Based on the table, we find that ICML achieves decent performance when train on noisy labels with noise level as high as 70%, and still able to achieve around 96% prediction accuracy at test time, which is significantly better than baseline (93%, see following discussions), and 1% lower than model trained with perfect labels. Loss is mostly consistent with test accuracy (reversely associated), but it varies more since we are using cross entropy loss which considers prediction probability, so even same prediction accuracy could have different loss associated.

Figure 7 captures ICML method's learning curve (i.e. prediction accuracy) for CNNs trained by the end of each training stage (epoch). We see that ICML method is able to learn on noisy labels, and we are not overfitting the network as we stops training at epoch where learning curve just becomes relatively flat, and test accuracy is still pretty high and close to validation set with most true labels recovered. Also, when we train on more epochs, it tends to learn very slowly going beyond 10 epochs.

Figure 8 computes and visualize confusion matrix based on ICML method predicted class output vs true classes. It shows that the network learns decently and is able to make prediction on test set with relatively high accuracy (the diagonal line has high number, meaning network making right class prediction most of the time and it is not getting 'confused').

Overall, ICML method achieves decent performance training on different noise levels, it is much higher than baseline but not close to oracle or state-of-art method on its ow. Intuitively, it is performing majority vote ensemble

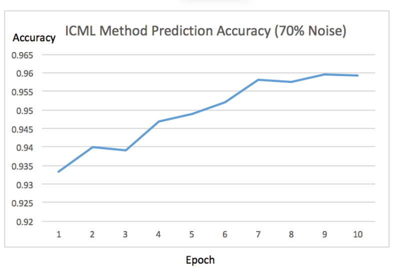
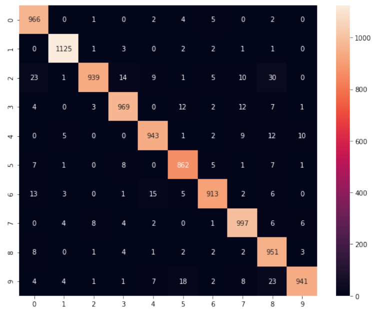


Figure 7. ICML Method Learning Curve: per epoch accuracy (70% noise)



	0	1	2	3	4	5	6	7	8	9
0	966	0	1	0	2	4	5	0	2	0
1	0	1125	1	3	0	2	2	1	1	0
2	23	1	939	14	9	1	5	10	30	0
3	4	0	3	969	0	12	2	12	7	1
4	0	5	0	0	943	1	2	9	12	10
5	7	1	0	8	0	862	5	1	7	1
6	13	3	0	1	15	5	913	2	6	0
7	0	4	8	4	2	0	1	997	6	6
8	8	0	1	4	1	2	2	2	951	3
9	4	4	1	1	7	18	2	8	23	941

Figure 8. ICML Method Prediction Confusion Matrix

Noise level	Test Accuracy	Final Epoch Training Loss
30%	0.9833	2.0429
50%	0.9740	2.1151
70%	0.9657	2.1330

Table 2. ICL-R Method Accuracy & Loss trained on Noise Levels 30%, 50%, and 70%

learning at each epoch so CNNs can share the learning and cross checking, and it converges relatively fast with ICML label update. However, when all label predictions from 3 CNNs are wrong (this is especially common when noise level is high and at early traing stages), majority vote is not that helpful and simply stucks at wrong label. To address this issue, we need to introduce randomness.

5.3. Iterative Cross Learning - Random (ICL-R)

To overcome ICML's drawback, we then experiment ICL-R training approach on noisy labels by introducing randomness in label update. For each run, instead of updating label based on majority vote from CNN outputs, we assign random label to data if CNN outputs disagree. By running training ten times and averaging the results, its accuracy and loss trained on different noise levels (30%, 50%, and 70%) are captured in Table 2.

We see that the accuracy (e.g. 96.57% in 0.7 noise level) is higher than that of ICML approach, since introducing ran-

Noise level	Test Accuracy	Final Epoch Training Loss
30%	0.9864	2.0110
50%	0.9810	2.1164
70%	0.9709	2.1326

Table 3. ICML+ICLR Hybrid Label Update Method Accuracy & Loss trained on Noise Levels 30%, 50%, and 70%

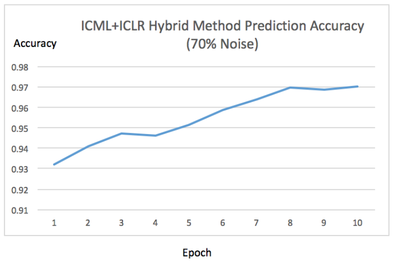


Figure 9. ICML+ICLR Hybrid Method Learning Curve: per epoch accuracy (70% noise)

domness reduce the impact of stucking in wrong labels during training and helps model learn. However, it typically takes longer to train and to converge, due to the same reason - random labeling when model disagrees might keep updating the label with different class and thus network learning speed is relatively slower.

5.4. ICML + ICL-R: Hybrid Label Update

Based on observations in the experiments above, we propose a new ICML+ICLR hybrid label update method, which is to perform random update only when CNN output all disagrees, otherwise perform majority vote when there is a majority. By running training ten times and averaging the results, its accuracy and loss trained on different noise levels (30%, 50%, and 70%) are captured in Table 3.

Based on the table, we find that this method outperforms the previous ones as expected, and reaches near state-of-the-art test time prediction accuracy 97% even trained on 70% noisy dataset. Confusion matrix in Figure 10 also supports this observation - more instances are predicted correctly across 10 classes. We also plot the learning curve and the model does not appear to be overfitting/underfitting, as we stop when curve went flat, and test accuracy is high and close to validation accuracy with mostly recovered labels from ICML+ICLR label updates.

We think this is the best model as it is simple, generaized (no loss function design, no data assumption, directly learn on noisy labels with high noise level), and achieved close to SOTA result with minimum tuning effort. It combines the strength of ICML and ICLR methods that it learns and converge fast with majority vote, and able to avoid being stuck in wrong label update by introducing randomness.

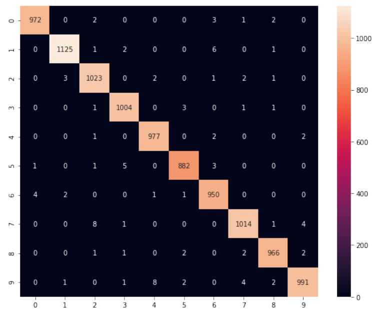


Figure 10. ICML+ICLR Hybrid Method Prediction Confusion Matrix

Noise level	Test Accuracy	Final Epoch Training Loss
30%	0.9845	2.0672
50%	0.9773	2.1058
70%	0.9695	2.1776

Table 4. ICML+ICLR Multi-stage Learning Method Accuracy & Loss trained on Noise Levels 30%, 50%, and 70%

5.5. ICML + ICL-R: Multi-stage Learning

We also experiment using ICL-R at early training stages (epochs) and ICML in later stages. Table 4 captures the result, it also achieves good performance (96.9% with 70% noise), but is slightly weaker than pure hybrid update method, which makes sense since this method is essentially a special case of hybrid label update, instead of forcing ICML update in later stage and ICLR update in early stage, the previous hybrid update approach dynamically decides which strategy to use based on label outputs and adjust update during training, so it is more generalized and flexible, and should achieve slightly better result than multi-stage learning as is shown in our experiments.

5.6. Baseline and Oracle

To evaluate the effectiveness of proposed approach, we compare them with a baseline model (lower bound estimate) and oracle model (upper bound estimate). Baseline model is a tuned CNN trained on the corrupted dataset directly, while Oracle model is a tuned CNN trained on original true label dataset (no noise introduced). Here we train a 3 layer CNN using conv-relu-conv-relu-maxpool-fc-dropout-softmax with dropout on noisy labels with no label update, and on true label data respectively. Table 5 shows test time error rate for baseline model (e.g. 6.76% with 70% noise) and oracle models (1.25%).

We summarize experimentation results in Table 6. We find that ICML+ICLR hybrid update is the best approach to address noisy label problem, which reaches near SOTA test accuracy (over 97% on 70% noise) that is much higher than baseline and only 1.5% lower than oracle accuracy.

Noise level	Baseline Test Error	Oracle Test Error (%)
30%	2.29	1.25 (no noise*)
50%	3.78	1.25 (no noise*)
70%	6.76	1.25 (no noise*)

Table 5. Baseline and Oracle Model Test Time performance

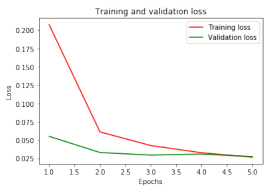


Figure 11. Perfect Label CNN Training Loss

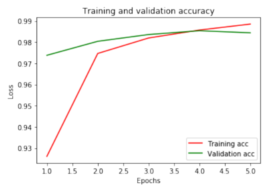


Figure 12. Perfect Label CNN Training Accuracy

Noise level	0.3	0.5	0.7
Baseline	0.9771	0.9622	0.9324
Oracle	0.9875	0.9875	0.9875
ICML	0.9791	0.9695	0.9596
ICL-R	0.9833	0.9740	0.9657
ICML+ICLR hybrid*	**0.9864**	**0.9810**	**0.9709**
ICML+ICLR staged	0.9845	0.9773	0.9695

Table 6. Experimentation Summary: average test set prediction error rate on 10 runs of our proposed methods, where ICML+ICLR hybrid update method* appears to be most effective for learning on noisy labels, which surpasses state of art test time prediction accuracy in some cases [21, 5] with little tuning.

5.7. CNN Visualization Analysis

We visualized conv layer outputs from model trained on true label and noisy label respectively, from Figure 13 and 14, we can see that model trained on noisy labels do tend to learn/recognize features less clearly compared to true label case. In Figure 15, we show a mis-classified image sample where true label is 2 while our model predicts 7. From the conv layer output, it appears that the features are vaguely learned and many of them appears similar to 7 visually due to lack of detail/clarity learned by CNNs.

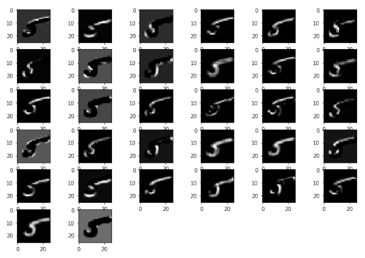


Figure 13. Visualizing Conv layer output: learning with true label

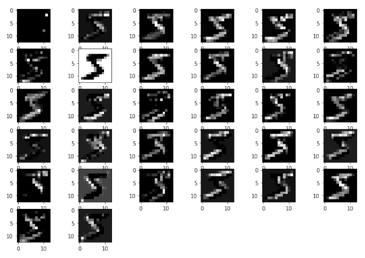


Figure 14. Visualizing Conv Layer output: learning with noise label

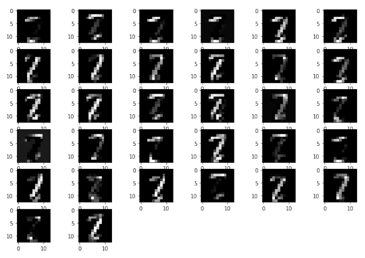


Figure 15. Sample mis-classified data at test time

6. Conclusion/Future Work

This paper proposes to train multiple (> 3) CNNs in parallel and perform a hybrid update strategy with majority vote and random label update based on CNNs outputs for training data. Test time performance proves the effective of this method, which is able to achieve near state of the art 97% accuracy with noise level as high as 70% in training set. It is a generalized, flexible, and performant method that can serve as a base method to plug in other noise label learning techniques for better performance, and can also be used to automatically label data by assigning random label and training CNNs with this method on these data along with a portion of true label data provided to perform label update.

For future work, we want to test method effectiveness on more complicated images such as Cifar10 [10], Fashion MNIST [23], better tune the network, and seek theoretical proof for convergence analysis.

7. Contributions & Acknowledgements

Although this is a solo project, I leverage CS231N starter code for tensorflow CNN (Code link: http://cs231n.github.io/assignments/2019/spring1819_assignment2.zip) and follows advice on CS231N notes/materials extensively, such as CNN architecture, hyper-parameter tuning, etc. I also appreciate the student credits provided by Stanford for me to speed up training on GPUs with Google Cloud Compute Engine, as well as valuable advice/feedback provided by Stanford TAs/Faculties throughout the course.

Note1: I intend to polish results, add theoretical prove and submit this work as a paper to computer vision conference.

Note2: code (to be cleaned up): https://github.com/honghaoq/Noise-CNN-Addressing-Noisy-Label-Problem-with-Convolutional-Neural-Network (key codes mainly in this iPython notebook: https://github.com/honghaoq/Noise-CNN-Addressing-Noisy-Label-Problem-with-Convolutional-Neural-Network/blob/master/Noise_CNN.ipynb)

References

[1] T. R. A. Rasmus. Semi-supervised learning with ladder networks, 2015. Advances in Neural Information Processing Systems `pages 3546-3554`.

[2] R. Barandela and E. Gasca. Decontamination of training samples for supervised pattern recognition methods. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 621–630. Springer, 2000.

[3] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[4] 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.

[5] Y. Bodi, J. Chen, W. Zhang, H.-S. Tai, and S. McMains. Iterative cross learning on noisy labels. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 757–765. IEEE, 2018.

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

[7] T. G. Dietterich. Ensemble methods in machine learning, 2000. MCS '00 Proceedings of the First International Workshop on Multiple Classifier Systems `pages 1-15`.

[8] Y. B. I. Goodfellow. Generative adversarial nets, 2014. Advances in neural information processing systems `pages 2672-2680`.

[9] X. I.Jindal. Learning deep networks from noisy labels with dropout regularization, 2016. 2016 IEEE 16th International Conference on Data Mining `pages 967-972`.

[10] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[11] 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.

[12] J. Larsen, L. Nonboe, M. Hintz-Madsen, and L. K. Hansen. Design of robust neural network classifiers. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, pages 1205–1208. IEEE, 1998.

[13] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[14] 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.

[15] A. Levin, P. A. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *ICCV*, volume 1, page 2, 2003.

[16] V. Mnih and G. E. Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML-12)*, pages 567–574, 2012.

[17] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Cikm*, volume 5, page 3, 2000.

[18] 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.

[19] 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.

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

[21] X. W. T. Xiao. Learning from massive noisy labeled data for image classification, 2015. 2015 IEEE Conference on Computer Vision and Pattern Recognition `pages 2691-2699`.

[22] C.-M. Teng. A comparison of noise handling techniques. In *FLAIRS Conference*, pages 269–273, 2001.

[23] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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