A Simple Approach to the Noisy Label Problem Through the Gambler's Loss

Anonymous authors
Paper under double-blind review

Abstract

Learning in the presence of label noise is a challenging yet important task. It is crucial to design models that are *robust* to noisy labels. In this paper, we discover that a new class of loss functions called the gambler's loss provides strong robustness to label noise across various levels of corruption. Training with this modified loss function reduces memorization of data points and is a simple yet effective method to improve robustness and generalization. Moreover, using this loss function allows us to derive an analytical early stopping criterion that accurately estimates when over-memorization begins to occur. Our overall approach achieves strong results and outperforming existing baselines.

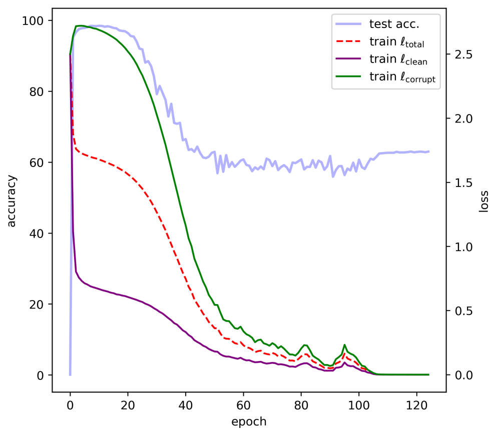
1 Learning from Noisy Labels

As a first step of supervised learning tasks, the user often collects a large amount of data in the form of (x, y) pairs where x represents the input data and y the desired labels. However, parts of world data can often be mislabeled due to 1) annotator mistakes as a natural consequence of large-scale crowdsourcing procedures (Howe, 2008), 2) the difficultly in fine-grained labeling across a wide range of possible labels (Russakovsky et al., 2015), 3) subjective differences when annotating emotional content (Busso et al., 2008), and 4) the use of large-scale weak supervision (Dehghani et al., 2017). Learning in the presence of noisy labels using neural networks is challenging since neural networks are overparametrized and are known to be able to memorize all labels even in the presence of strong regularization (Zhang et al., 2017). When the model memorizes the noisy labels, its generalization deteriorates (e.g., see Figure 1 and Section 1.1). In fact, this agrees with the theoretical finding in (Nakkiran et al., 2019) that neural networks trained with SGD often learns functions stage by stage from simpler functions to more complex ones, and functions whose output is uniformly random are the functions with highest complexity, and so memorization of random labels often late in the training. Based on these, we may devise special methods to learn signal from noisy labels. Similar to Patrini et al. (2017); Yu et al. (2019); Han et al. (2018), we deal with an idealized noisy label setting, where a noise transition matrix is invoked to define the problem, and each label has the probability defined by the transition matrix to be flipped to another label (see appendix for a formal mathematical definition).

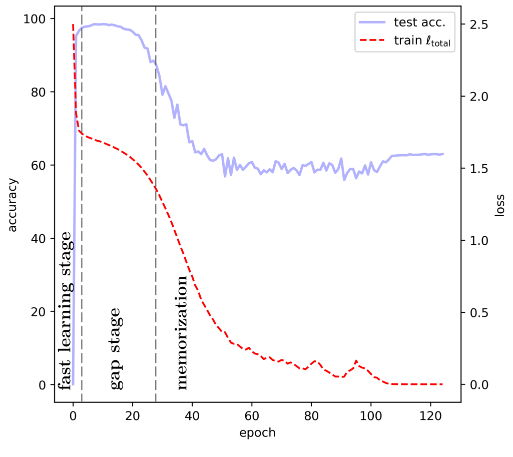
Organization: In this section, we first hypothesize and then empirically verify the existence of three stages during training in the presence of noisy labels; in Section 2, we propose that the gambler's loss function is a *noise robust loss function* that can be used for noisy label problems, and we further propose an *analytical early stopping criterion* using the gambler's loss. We then verify our methods through a series of experiments and comparisons with existing approaches.

1.1 Training Trajectory of Noisy Labels

In practice, one often faces the problem of learning from a dataset $\mathcal{D} = \{x, y\}_{i=1}^{n}$ whose labels are partially corrupted, i.e. $\mathcal{D} = \mathcal{D}_{clean} \cup \mathcal{D}_{\text{corrupt}}$. We consider a setting where the labels are symmetrically corrupted (i.e. any label has equal probability to be flipped to any other class). Let $\epsilon = 1 - r$ denote the probability that a data point (x, y) is clean. A classification problem can be denoted as learning a function $f(\cdot) : X \to Y$, where $x \in X = \mathbb{R}^n$ are the data points, and $y \in Y = \mathbb{R}^m$ are the targets. In practice, we parametrize f with parameters θ (i.e. f_θ) and learn θ by minimizing the empirical risk $\mathbb{E}[\ell(f_\theta(x), y)]$ for some loss function ℓ. Traditional learning methods perform gradient updates on θ by minimizing the empirical loss over the entire dataset $\mathcal{D}$. However, naive training on $\mathcal{D}$ often results in a "peaked" learning curve as shown in Figure 1b. We observe that such learning curves seem to consist of three stages (See Figure 1):



(a)



(b)

Figure 1: Different stages in the presence of label noise. (a) We plot ℓ_{total} (total loss), ℓ_{clean} (loss on $\mathcal{D}_{\text{clean}}$), and ℓ_{corrupt} (loss on $\mathcal{D}_{corrupt}$), we see that during the gap stage, there is a clear "gap" between the ℓ_{clean} and ℓ_{corrupt} where training on clean labels has completed but training on noisy labels has barely started; (b) Hypothesized qualitative division of the three stages: fast learning stage, gap stage, and the memorization stage. Experiment done on MNIST with corruption rate 0.5.

1) **Fast Learning Stage**: The model *quickly* learns the underlying mapping from data to clean labels; one observes rapid decreases in training loss and increases in test accuracy. Compare Figure 1a with 1b.

2) **Gap Stage**: From Figure 1a, we notice learning on the clean labels is almost complete ($\ell_{clean} \sim 0.5$) but training on noisy labels has not started yet ($\ell_{\text{corrupt}} \sim 2.5$), and a large gap in training loss exists between $\mathcal{D}_{clean}$ and $\mathcal{D}_{\text{corrupt}}$. Both the train loss and the test accuracy reach a plateau, and this is the time at which the generalization performance is the best.

3) **Memorization**: When there are only noisy labels left to learn from, the model will *memorize* these noisy labels and the train loss decreases slowly to 0. However, memorizing noisy labels *hurts generalization* and the test accuracy decreases as well.

These stages seem to be present across datasets and model architectures when label noise is present. This is problematic because the testing accuracy reaches a peak early in training and then decreases steadily because the noisy gradient from $\mathcal{D}_{\text{corrupt}}$ starts to dominate that from $\mathcal{D}_{clean}$, leading to memorization of noisy labels and affecting generalization. We verified that . Therefore, tackling the problem of learning from noisy labels involves reducing the negative effect from $\mathcal{D}_{\text{corrupt}}$. This is hard in general, since one does not know a priori which points are mislabeled. We mainly deal with symmetric noise, and we denote the corruption rate (the probability that a data point has a label that is not its true label) as $r = 1 - \epsilon$, where ϵ is the clean rate. Various approaches have been proposed (Patrini et al., 2017; Han et al., 2018; Yu et al., 2019), but, unfortunately, the best current methods can do is only to slow down the drop in test accuracy instead of preventing it completely (Han et al., 2018).

Our method will be based on the above observation; to summarize the key point, we note:

Observation. *When training on a label-corrupted dataset, there is a stage when the average loss on the correct points is much lower that of the corrupted points.*
Intuitively, since features are easier to learn than random noises on average. This phenomenon also motivates a label noise method sometimes called the "small loss method" (Han et al., 2018; Yu et al., 2019), in which the data points with loss higher than a threshold is screened to prevent the network from learning wrong information from such point. The observation also holds theoretically. As shown in (Nakkiran et al., 2019), neural networks trained with SGD often learns functions stage by stage from simpler functions to more complex ones, and functions whose output is uniformly random are the functions with highest complexity. From this observation, we make the following idealized assumption that our method is based on:

Assumption. *(Idealized gap stage) During the gap stage, the model has not learned anything about the data points in $\mathcal{D}_{\text{corrupt}}$, and predicts strictly uniform score across all the labels, and it achieves perfect accuracy on $\mathcal{D}_{clean}$.*

We find that this assumption holds well for simple datasets such as MNIST and on datasets with very high corruption rate, where our method achieves best results, and less so on more complicated datasets such as CIFAR10. However, the key point is that it always holds to some degree. We next show that training classifier with the gamblers loss term (Ziyin et al., 2019) increases the gap for easier identifiability and slows down the memorization phenomenon.

2 The Gambler's Loss Function and How to Early Stop

We propose an approach that uses the gambler's loss function. Intuitively speaking, the gambler's loss builds on the analogy that making a classification given a data point is the same as a gambler making a bet on a horse race given the horses, and that minimizing the nll loss in the classification problem is the same as maximizing the doubling rate of wealth. The gambler's loss maybe generalized to include a reservation option in which the gambler can play "safe" and reserve part of his money in pocket. We refer to this generalized version as the *gambler's loss*. Mathematically, the gambler's loss augments the target space by an "reservation" dimension: $Y \to Y' = \mathbb{R}^{m+1}$, where the $(m+1)$-th dimension is defined as the reservation score. In the case of neural network classifier with an output logistic classifier, this involves adding one more output neuron. We suppose outputs of a neural network are normalized to form a distribution. The gambler's loss function for a single data point (x, y) with y being a point mass, and with neural network $f_{\mathbf{w}}(\cdot) : X \to Y'$ is

$$\ell(f(x), y) = \log\left(f(x)_y + \frac{1}{o} f(x)_{m+1}\right) \quad (1)$$

In more general settings, the target may also be a distribution, but should not undermine the arguments in this paper. o is a hyperparameter for the loss function, and reasonable range of o is $o \in (1, m]$. Larger o encourages reservation.

Traditional methods in label noise often involves introducing a surrogate loss function that is specialized for the corrupted dataset at hand and is of no use when one is not aware of the existence of label noise (Patrini et al., 2017). Therefore, one surprising finding in this work is that simply using the gambler's loss provides robustness to label noise *automatically* (see Section 4.1), whereas the gambler's loss function is a very general loss function and is noise-agnostic. In particular, it shows that the gambler's loss function makes learning on corrupted points slower while not affecting the clean points so much. As a result, this widens the test accuracy plateau by slowing down the memorization phenomenon. In some cases, it even leads to a better convergence speed and better peak performance (see Figure 4). This may be justified by the following argument. Notice that the gambler's loss function has derivative:

$$\frac{\partial \ell(f_\theta(x))}{\partial \theta} = \frac{1}{f(x)_y + \frac{1}{o} f(x)_{m+1}} \frac{\partial f_\theta(x)}{\partial \theta} \quad (2)$$

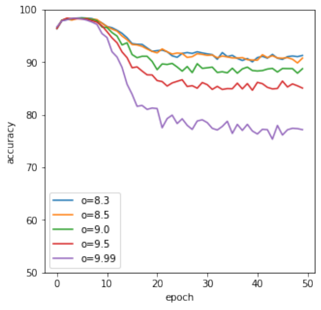
and by Theorem 1 (in Section 2.2), during the gap stage we have (j being the true label):

$$\frac{1}{f(x)_y + \frac{1}{o} f(x)_{m+1}} = \begin{cases} \frac{1}{\epsilon}, & \text{if } y = j \\ \frac{o-1}{1-\epsilon}, & y \neq j \end{cases} \quad (3)$$

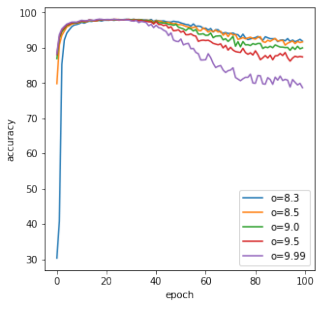
and the probability we update a clean data point is ϵ, while that of a corrupted data is $\frac{1-\epsilon}{M-1}$, and, therefore, in expectation (sloppy notation), the gradient is of order

$$\mathbb{E}\left[\frac{\partial \ell(f_\theta(x))}{\partial \theta}\right] \sim \frac{\partial f_\theta(x)}{\partial \theta} \qquad (\text{if } y = j) \quad (4)$$

$$+ \frac{o-1}{M-1} \frac{\partial f_\theta(x)}{\partial \theta}, \qquad (\text{if } y \neq j) \quad (5)$$

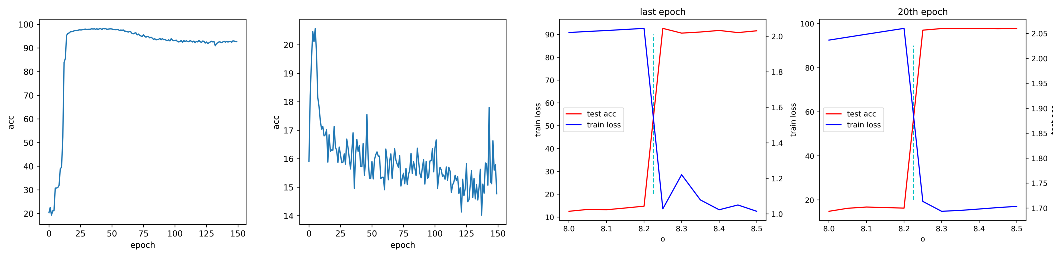


(a) Training with Adam



(b) Training with SGD

Figure 2: Training trajectories on the gambler's loss using different optimizers, with $r = 0.5$.



(a) testing accuracy at $o = 8.25$ (b) testing accuracy at $o = 8.20$ (c) critical behavior around $o = 8.2$ at $epoch = 20$ (d) critical behavior around $o = 8.2$ at $epoch = 200$

Figure 3: Critical behavior of the gambler's loss. Data showing that the learning almost do not happen at all for $o < o_{\text{crit}}$, while above o_{crit} the behavior is qualitatively similar. The optimal o can be tuned for using this phenomenon; however, more often one does not need to tune for o.

In general, $o - 1 < M - 1$ and so a corrupt data, in expectation, has gradient $\frac{o-1}{M-1}$ times smaller than that of the clean data, and, within a proper range of o, the lower the o is, the slower the model learns the corrupt labels, providing stronger robustness. Intuitively speaking, this argument means that *making random bet will help with making money, and a "skilled" gambler will not make such bet*. This trend is what we observe in Section 4.1 when o lies above a critical value. On the other hand, this analysis suggests that, if we train the gambler's loss with a subgradient methods such as Adam (Kingma & Ba, 2014), the constant factor in front of the gradient would be canceled by the preconditioner $1/\sqrt{\mathbb{E}g^2}$, and so the widening effect would not be observed. See Figure 2. The time it takes for Adam to reach and to start exiting the plateau seem invariant to o, while using SGD, we see that the plateau is widened as we decrease o. For Adam, even if the widening effect is not changed, we see that at convergence, smaller o does lead to higher performance using both optimizers. More plots and experiments using Adam are given in the appendix.

2.1 Tuning for the Optimal o and First-Order Phase Transition

We note that there is a "good" range for hyperparameter o and a bad range. The good range is between a critical value o_{crit} and M (non-inclusive) and the bad range is smaller than o_{crit}. We conducted substantial experiments to observe that any o within the good range are seen to provide improved robustness against label noise, and so most of the cases we do not tune for o but only to make sure it is within the good range. Moreover, tuning for the optimal o is fast and can be done in linear time, and we discuss this and its connection to the physical first-order phase transition theory in the appendix.

2.2 Early Stopping Criterion For the Gambler's Loss

Given the broadening of the gap and slowing down of memorization, the next steps is to accurately estimate the end of the gap and perform early stopping at this peak in test accuracy. This is shown to be true theoretically in Li et al. (2019) for overparametrized networks trained with gradient descent. However, it is surprising that the existing literature only focuses on delaying or relieving the negative effect of learning the noisy labels, but not on finding the best time to do early-stop (Patrini et al., 2017; Han et al., 2018; Yu et al., 2019). As in the previous works Han et al. (2018); Yu et al. (2019), we assume that the ϵ is known; otherwise, it may be estimated.

Let o be the gambler's hyperparameter. Let $\hat{p}$ be the predicted probability on the true label y, and let $\hat{k}$ denote the prediction made on all the wrong classes added altogether, $\hat{l}$ be the predicted confidence score by the gambler's loss. By definition of a probability distribution, we have $\hat{p} + \hat{k} + \hat{l} = 1$. Based on our observations, we make the following idealized assumption: during the the gap stage, the model has not learned anything about the data points with wrong labels, i.e., it makes strictly uniformly random guess across the all labels, the predicted probabilities are the same and are all equal to $\frac{\hat{k}}{M}$. This allows us to write down the general loss function during the gap stage:

$$\mathbb{E}[-\tilde{\ell}] = \epsilon \log\left(\hat{p} + \frac{\hat{k}}{M} + \frac{1 - \hat{p} - \hat{k}}{o}\right) + (1 - \epsilon) \log\left(\frac{\hat{k}}{M} + \frac{1 - \hat{p} - \hat{k}}{o}\right) \tag{6}$$

where the expectation is taken over the dataset, and we will omit it since it does not affect clarity. The following theorem uses the above assumptions and predict the loss of the gap stage.

Algorithm 1 Early Stopping on the Gambler's Loss

```
1: Given: untrained classifier parameters $\theta$, noise rate $\epsilon$, gambler's hyperparamter $o$.
2: Change classifier output from $m$-way to $(m+1)$-way softmax.
3: Compute early stopping criterion: $-\tilde{\ell}^*(\epsilon, o)$. If satisfied, end training.
4: for $(x, y)$ in each batch do
5:     Compute gambler's loss $\mathcal{L}_g$ from equation 1.
6:     If $\mathcal{L}_g < -\tilde{\ell}^*(\epsilon, o)$, end training.
7:     Update parameters $\theta$ using $\nabla_\theta \mathcal{L}_g$.
8: end for
9: return trained classifier parameters $\theta$.
```

Theorem 1. *During the idealized gap stage, the optimal solution has $\hat{k} = 0$, and so the loss function takes the form:*

$$-\tilde{\ell}(\hat{p}) = \epsilon \log\left(\hat{p} + \frac{1-\hat{p}}{o}\right) + (1-\epsilon)\log\left(\frac{1-\hat{p}}{o}\right) \tag{7}$$

which exhibits an optimal solution at

$$-\tilde{\ell}^*(\epsilon, o) = \min_p -\tilde{\ell}(p) = \epsilon \log \epsilon + (1-\epsilon)\log\left(\frac{1-\epsilon}{o-1}\right), \tag{8}$$

which only depends on ϵ and o, and we have $p^* = \frac{\epsilon o - 1}{o-1}$.

The proof is given in the appendix. This theorem tells us that a network trained with the gambler's loss function with hyperparameter o on a symmetrically corrupted dataset with corruption rate $1-\epsilon$ should have training loss around $-\tilde{\ell}^*(\epsilon, o)$ when entering the gap stage. This motivates using $-\tilde{\ell}^*(\epsilon, o)$ as the early stopping criterion. Notice that we expect our loss to have the following limits:

1. As $\epsilon \to 1$, we recover the no-noise setting, and the optimal loss is 0.

2. As $\epsilon \to 0$, we go to the full noise case, and the dis-confidence score would dominate, and the gambler's loss converge to $\log \frac{1}{o}$ (when o is large) (Ziyin et al., 2019).

As shown in Yu et al. (2019); Han et al. (2018), simply training on the raw loss functions features a gap stage that lasts only very shortly (~ 1 epoch), while the gambler's loss makes the network stays around its performance peak for a much longer time. The performance on the gambler's loss has barely dropped by the time the baseline almost dropped to random accuracy, and this makes it easy to detect the peak and do early-stopping. We summarize our proposed method in Algorithm 1.

3 Related Work

Label Noise: Modern dataset often contains a lot of labeling errors (Russakovsky et al., 2015; Schroff et al., 2010). Existing methods often focus on using a surrogate loss function (Patrini et al., 2017) that is specific to the label noise problem at hand, or design a special training scheme to alleviate the negative effect of learning the data points with wrong labels (Han et al., 2018; Yu et al., 2019). In this work, we mainly compare with the following methods. **F-Correction (FC)** Patrini et al. (2017): this method smooths the hard labels with the noise transition matrix; we estimate the transition matrix as suggested by the paper; **Co-teaching (CT)** (Han et al., 2018): this method trains two networks simultaneously, and update each with the other's predicted label to decouple the mistakes; **Co-teaching+ (CT+)** (Yu et al., 2019): this method improves the previous method by only propagating on the labels the two networks disagree. Also, we use the subscript **ES** as a shorthand for early stopping. Since the early stopping criterion for these methods are not clear, we early stop at where the training accuracy is larger than ϵ for comparison, since this is when the learning on the noisy data has definitely started.

Early Stopping. Early stopping is an old problem in machine learning Prechelt (1998); Amari (1998), and studying this in the setting of label noise also appeared recently Li et al. (2019); Hu et al. (2019). In particular, we propose to early stop on a loss function called the gambler's loss function Ziyin et al. (2019). Our analysis on this loss function allows us to propose a very simple analytic function to predict a early stopping threshhold without using a validation set and is independent of the model (as long as it is a neural network with enough capacity) or the task. Early stopping without validation set is also recently studied, but does not relate to the label noise problem (Mahsereci et al., 2017). It has been long noticed (Frénay & Verleysen, 2013; Han et al., 2018) and recently theoretically proven (Li et al., 2019) that early stopping can an effective way to defend against label corruption (but no actual method or heuristic for early stopping is given); to the best of our knowledge, this is first early stopping criterion to deal with the problem.

4 Experiments

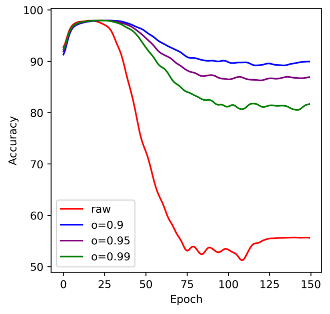
We design several experiments to evaluate 1) the robustness of gambler's loss to noisy labels, 2) the accuracy of our early stopping criterion as compared to existing heuristics, and 3) the performance of gambler's loss when applied to existing methods to combat noisy labels.

4.1 Robustness of Gambler's Loss to Label Noise

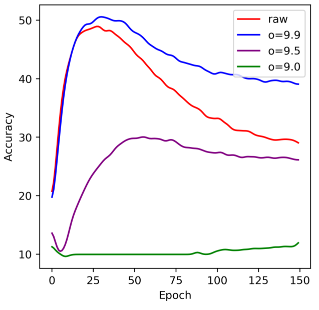
We first demonstrate that simple training on the gambler's loss provides robustness against label noise (also see Section 2.1). For this and later sections, the exact experimental details are given in the Appendix. For demonstration, we choose a simple CNN with 2 convolutional layers followed by 2 fully connected layers. See Figure 4. We see that with properly tuned o, the gamblers's loss alleviates the drop in performance caused by learning the corrupt labels. For example, in Figure 4a, all three choices of o makes the accuracy plateau wider than training on the raw loss (nll loss). At convergence, the final accuracy is much higher (~ 90%) than the baseline (~ 55%). For the experiments on CIFAR10, the results are slightly subtler. Here we see the effect of not properly tuning o. As shown in (Ziyin et al., 2019), the proper choice ranges from 1 to M; higher o ($\approx M$) encourages learning while very low o (~ 1) slows down learning. We notice that for a different corruption rate, a different o_{crit} seem to exist. Also see Table 1 (category $Gbler$), we see that at convergence, training on gambler's loss achieves much higher accuracy than the baseline training on *nll* loss.

4.2 Early Stopping

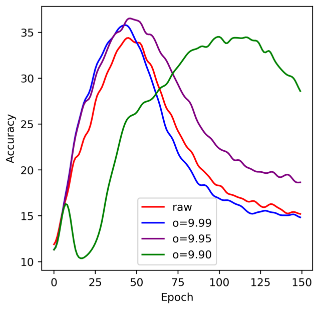
Now, we show that stopping at our predicted loss level succeeds in stopping at a point close to where the maximum accuracy is achieved. See Figure 5. We choose architectures of very different capacity. The experiment on MNIST is done as in the previous section and the experiment on CIFAR10 is done using ResNet-18. We also test on a wide range of noise levels ranging from small, with corruption rate 0.2, to extremely large, with $r = 0.85$ (the intermediate ranges with larger figures are given in the appendix, see Figure 7). We see that our method predicts close to optimal early stopping point for all corruption rates we tested on. Also notice how the learning curves are different for the two different datasets. This suggests that our theory works even in the presence of very different dynamics. The fact that this early stopping criterion works suggests in return suggests the validity of our assumption. In these experiment, we do not tune for optimal o, they are all set to $o = 9.9$; since the results are qualitatively similar when $o > o_{\text{crit}}$.



(a) MNIST: $r = 0.5$



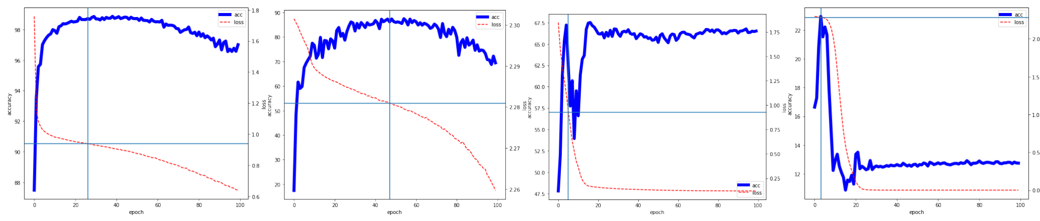
(b) CIFAR10: $r = 0.5$



(c) CIFAR10: $r = 0.8$

Figure 4: Testing accuracy through out training; raw refers training on the default loss function, i.e. the *nll* loss, raw is plotted in red. We see that, when $o > o_{\text{crit}}$, the gambler's loss provides robustness on its against noisy labels.

See Table 1 and also compare with other baselines in Table 2. We notice that doing this already achieves SOTA results on MNIST for all symmetric noise categories according to the results in Han et al. (2018) and Yu et al. (2019). Early stopping (AES) using our criterion is compared with a very standard early stopping method using validation set: we split 6000 images from the training set to make a validation set, and we early stop when the validation accuracy stops to increase for 5 consecutive epochs. There are also a few other validation-based early stopping criterion, but they



(a) MNIST: $r = 0.2$ (b) $r = 0.85$ (c) CIFAR10: $r = 0.2$ (d) $r = 0.85$

Figure 5: Early stopping on MNIST (1st row) and CIFAR10 (2nd row). r refers to corruption rate. The horizontal line is the predicted early stopping point. We see that this point corresponds to where the testing accuracy (blue solid line) is at maximum.

Table 1: Robustness of the Gambler's loss to noisy labels. We see that simply using gambler's loss on label noise problems improve. For the two categories using early stopping, the number in parenthesis refers to the avg. number of epochs to stop. Using gambler's loss stops at better accuracy, has smaller variance, and takes much shorter time.

Dataset	Performance Comparisons			Early Stopping Comparisons	
	nll loss	Gblers	Gblers + AES	Gblers + VES	Gblers + AES
MN $r = 0.2$	84.7 ± 0.5	96.7 ± 0.2	$\mathbf{98.8} \pm \mathbf{0.0}$	95.1 ± 0.4 (95)	$\mathbf{98.8} \pm \mathbf{0.1}$ (**17**)
MN $r = 0.5$	55.1 ± 3.1	91.2 ± 0.7	$\mathbf{98.0} \pm \mathbf{0.1}$	79.7 ± 2.0 (115)	$\mathbf{98.0} \pm \mathbf{0.0}$ (**18**)
MN $r = 0.65$	39.7 ± 2.5	85.9 ± 1.1	$\mathbf{97.1} \pm \mathbf{0.1}$	58.9 ± 2.5 (115)	$\mathbf{97.2} \pm \mathbf{0.2}$ (**15**)
MN $r = 0.8$	19.1 ± 3.0	76.1 ± 0.3	$\mathbf{92.3} \pm \mathbf{0.6}$	21.1 ± 0.1 (117)	$\mathbf{93.5} \pm \mathbf{0.3}$ (**15**)
MN $r = 0.85$	14.5 ± 0.7	71.0 ± 1.4	$\mathbf{86.3} \pm \mathbf{0.5}$	15.0 ± 1.1 (110)	$\mathbf{85.2} \pm \mathbf{1.7}$ (**13**)

are shown to perform qualitatively and quantitatively similar (Prechelt, 1998), so we only compare with this method. This early stopping is called **VES** (validation early stopping) and our method is called **AES** (analytical early stopping). We see that our proposed method significantly outperforms the baseline early stopping method; this is because a small validation set may have large variance and this problem is more serious when label noise is present (since validation sets are often split from the train set). In the VES vs. AES comparison, we fix $o = 9.99$ and is not directly comparable to the left parts in the table. In the left part, We also notice that training and stopping on the gambler's loss is especially effective on tasks with extreme corruption rate ($r = 0.85$), with around 70% accuracy improvement over the baseline and 28% improvement over the runner-up SOTA methods.

4.3 Combining Gambler's Loss with Other Methods and Benchmarking

We also show how our method might be combined with SOTA methods (we choose CT). To combine gambler's loss with co-teaching, we first replace the loss function in Co-Teaching with the gambler's loss, and reject fewer than or equal to ϵ data points by sorting on the original loss function from the largest to smallest. This modification only shows how we might combine our method with existing methods, and we do not argue that it is the optimal for such combination. We call our modified version Co-Gambler, and specific details of the modification is given in the Appendix; we also simplify CT by eliminating *four* of its hyperparameters. Namely, we do not use learning rate and momentum schedules, and we use not use drop rate schedule for co-Gambler (while these are used for the CT columns). See Table 2. We see that on MNIST, combining our method with Co-Teaching always outperforms the baselines. On CIFAR10, our method seems to work better when the noise level is extreme ($r \sim 0.7, 0.8$), agreeing with our finding in the previous section. For the CIFAR10 experiments, we do not compare with CT+ since it is a modified version of CT and we notice that it fails in the same range as CT ($r = 0.7, 0.8, 0.85$). We notice that this way of combining gambler's loss improves on pure AES on gambler's loss only by $3 - 4\%$, while the training time more than doubles with twice as many parameters. The reason for the limitation of this combination is simple: CT involves a special training procedure that throws away a significant percentage of the data, thus changing the data distribution. Our method, however, assumes that the data distribution is intact, and can be well described by ϵ. Early stopping on gambler's loss, running much faster than CT and its variants, still achieves SOTA results at $r = 0.85$, which shows the strength of our simple method. The method that has similar complexity and runs at similar speed as ours is FC; FC, however, requires knowing the whole transition matrix, and is outperformed significantly by our method.

Table 2: Improving Co-Teaching with Gambler's Loss. We see that on MNIST (MN), combining our method with Co-Teaching always outperforms the baselines. On CIFAR10 (CF10), our method achieves clear superiority when the noise level is extreme ($r \sim 0.7, 0.8$), agreeing with our finding in the previous section. (*NA means that the ES stopping point is not reached by the algorithm.)

Dataset	FC	CT	CT+	CT$_{ES}$	CT+$_{ES}$	Co-Gbler$_{AES}$
MN r = 0.2	98.8 ± 0.1	97.2 ± 0.0	97.8 ± 0.1	98.6 ± 0.2	97.6 ± 0.1	**98.9 ± 0.0**
MN r = 0.5	79.6 ± 2.0	93.3 ± 0.1	95.4 ± 0.2	97.6 ± 0.1	96.3 ± 0.2	**97.9 ± 0.0**
MN r = 0.65	29.1 ± 0.1	92.5 ± 0.1	92.5 ± 0.1	**97.3 ± 0.2**	95.6 ± 0.2	**96.7 ± 0.3**
MN r = 0.8	19.8 ± 0.1	78.1 ± 0.5	67.5 ± 0.1	82.1 ± 1.7	71.4 ± 0.3	**92.5 ± 0.5**
MN r = 0.85	11.8 ± 0.0	60.5 ± 0.2	10.1 ± 0.0	66.5 ± 0.5	NA	**88.7 ± 0.5**
	FC	CT	CT$_{ES}$	Co-Gbler$_{ES}$	Gblrs$_{AES}$	
CF10 r = 0.2	67.0 ± 0.3	**71.0 ± 0.6**	69.5 ± 0.4	68.9 ± 1.0	67.5 ± 1.0	
CF10 r = 0.5	52.2 ± 1	**65.2 ± 0.2**	64.5 ± 0.8	60.0 ± 1.2	59.5 ± 0.4	
CF10 r = 0.6	45.2 ± 0.5	61.1 ± 0.2	**62.0 ± 0.3**	56.6 ± 1.5	51.3 ± 0.5	
CF10 r = 0.7	33.2 ± 0.5	48.9 ± 0.2	**49.1 ± 1.0**	**50.1 ± 1.5**	48.2 ± 0.5	
CF10 r = 0.8	22.7 ± 0.3	25.0 ± 0.1	25.8 ± 1.0	**40.7 ± 2.0**	37.0 ± 0.7	
CF10 r = 0.85	16.7 ± 0.3	17.3 ± 0.1	NA	16.7 ± 0.7	**18.9 ± 1.0**	

5 Conclusion and Discussion

In this work, we have proposed a simple and effective method to deal with label noise problems. This paper promotes the gambler's loss as a noise robust loss function, shown to improve performance of existing classifier model, while the early stopping criterion provides a simple criterion for early stopping when the corruption rate is known. Depending on the problem, one may decide to use the complete framework together (gambler's loss training + early stopping), or, when the assumption is not well satisfied, only use the gambler's loss to improve the model's robustness against a label-corrupted learning task. This urges for an understanding for why the gambler's loss actually works. We plot in training accuracy for MNIST with corruption rate 0.8 in Figure 6. We see that the gambler's loss seems to work in a fundamentally different way from the standard *nll* loss. Training on *nll* loss achieves 100% accuracy on the training set, with its loss approaching 0; however, the gambler's loss does not do so, its training accuracy stops increasing at around 36%, which is slightly above ϵ. We note that this observation also holds for various corruption rates, and for different optimizers such as SGD. Comparing with Table 5, we see that the gambler's loss achieves about 76% testing accuracy, while the baseline drops to 19% at this time. The vertical line also shows where our early stopping criterion is reached, at this point is training accuracy is ~ 20% with testing accuracy ~ 92%. We hypothesize that the main reason for gambler's loss's outperforming the baselines is that it automatically chooses a subset of the training set to learn on, thus avoiding a large part of the dataset that is corrupted.

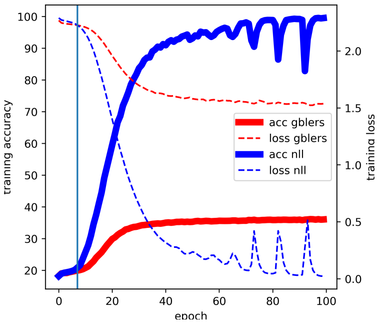


Figure 6: Training accuracy and training loss On MNIST with corruption rate 0.8 with Adam. o = 9.7. Training with gambler's loss prevents memorization of noisy labels. At convergence, *nll* loss reaches 19% testing accuracy, while the gambler's loss stays around 76%.

This result may provide insight into the current debate on the generalization-memorization effect (Zhang et al., 2017; Wu et al., 2017; Hu et al., 2019; Arpit et al., 2017). A key argument from Zhang et al. (2017) was "*deep neural networks easily fit random labels...; explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error*." Our results above, however, shows that this is not the case when a different loss function is chosen. While the *nll* loss encourages learning and memorizing the training set, the gambler's loss does not seem to do so. Our findings suggests that the key to understanding the memorization effect might be on the loss function itself, not the architecture of neural networks or explicit regularization, this is also what is suggested by Abiodun et al. (2018). We hypothesize that it may be seen as a natural generalization of the *nll* loss function, since it only involves adding a task independent term to the original *nll* loss, and using $o > o_{crit}$ seems always beneficial. We would like to study the memorization effect using the gambler's loss in more detail. Applications to real-world large-scale datasets with real label noise are also ripe avenues for future work.

REFERENCES

Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11):e00938, 2018.

Shun-Ichi Amari. Natural gradient works efficiently in learning. Neural Comput., 10(2):251–276, February 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL http://dx.doi.org/10.1162/089976698300017746.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, 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.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth Narayanan. Iemocap: interactive emotional dyadic motion capture database. Language Resources and Evaluation, 42(4):335–359, 2008. URL http://dblp.uni-trier.de/db/journals/lre/lre42.html#BussoBLKMKCLN08.

Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision. In arXiv, 2017. URL https://arxiv.org/abs/1711.00313.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. IEEE transactions on neural networks and learning systems, 25(5):845–869, 2013.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In Advances in neural information processing systems, pp. 8527–8537, 2018.

Jeff Howe. Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business. Crown Publishing Group, New York, NY, USA, 1 edition, 2008. ISBN 0307396207, 9780307396204.

Wei Hu, Zhiyuan Li, and Dingli Yu. Understanding generalization of deep neural networks trained with noisy labels. arXiv preprint arXiv:1905.11368, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014. URL http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14.

L.D. Landau and E.M. Lifshitz. Statistical Physics. Number v. 5. Elsevier Science, 2013. ISBN 9780080570464. URL https://books.google.co.jp/books?id=VzgJN-XPTRsC.

Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. arXiv preprint arXiv:1903.11680, 2019.

Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. Early stopping without a validation set. arXiv preprint arXiv:1703.09580, 2017.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. arXiv preprint arXiv:1905.11604, 2019.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 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.

Lutz Prechelt. Early stopping-but when? In Neural Networks: Tricks of the trade, pp. 55–69. Springer, 1998.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3):211–252, 2015.

Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Harvesting image databases from the web. IEEE transactions on pattern analysis and machine intelligence, 33(4):754–766, 2010.

L. Wu, Z. Zhu, and W. E. Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes. ArXiv e-prints, June 2017.

Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In International Conference on Machine Learning, pp. 7164–7173, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2017. URL `https://arxiv.org/abs/1611.03530`.

Liu Ziyin, Zhikang Wang, Paul Pu Liang, Ruslan Salakhutdinov, Louis-Philippe Morency, and Masahito Ueda. Deep gamblers: Learning to abstain with portfolio theory. arXiv preprint arXiv:1907.00208, 2019.

A Mathematical symbols

$$N: \textit{number of input dimension}$$
$$M: \textit{number of output dimension}$$
$$\mathcal{D}: \textit{dataset},\ |\mathcal{D}|: \textit{size of the dataset}$$

B Experiment Details

We use Pytorch as the framework of implementation. Code to this paper will be released at *https://*********

C Theorem proof

In this section we prove theorem 1. We first show that $\hat{k} = 0$. Intuitively speaking, this simply means that a gambler betting randomly will not make money, and so the better strategy is to reserve money in the pocket, and so it suffices to show that for any solution $\hat{p}, \hat{k}, \hat{l}$, the solution $\hat{p}' = \hat{p}, \hat{k}' = 0, \hat{l}' = \hat{l} + \hat{k}$ achieves better or equal doubling rate. For a mislabeled point (we drop $\hat{\cdot}$), the loss is $\log(\frac{k}{M} + \frac{l}{o})$ but $M > o$, and so $\log(\frac{k}{M} + \frac{l}{o}) < \log(\frac{k+l}{o})$, and we have that optimal solution always have $\hat{k} = 0$.

Now, we find the optimal solution to

$$-\tilde{\ell}(\hat{p}) = \epsilon \log\left(\hat{p} + \frac{1-\hat{p}}{o}\right) + (1-\epsilon)\log\left(\frac{1-\hat{p}}{o}\right) \tag{9}$$

by taking the derivative with respect to p:

$$-\frac{\partial \tilde{\ell}}{\hat{p}}(\hat{p}) = \epsilon\frac{o-1}{(o-1)\hat{p}+1} + (1-\epsilon)\frac{-1}{1-\hat{p}} \tag{10}$$

and then setting it equal to 0

$$-\frac{\partial \tilde{\ell}}{\hat{p}}(\hat{p}) = \epsilon\frac{o-1}{(o-1)\hat{p}+1} + (1-\epsilon)\frac{-1}{1-\hat{p}} = 0 \tag{11}$$

is the $\hat{p}_{optimal}$:

$$\hat{p}_{optimal} = \frac{\epsilon o - 1}{o-1} \tag{12}$$

and then plugging into the original equation [9]:

$$-\tilde{\ell}^*(\epsilon, o) = \min_p -\tilde{\ell}(p) = \epsilon\log\epsilon + (1-\epsilon)\log\left(\frac{1-\epsilon}{o-1}\right) \tag{13}$$

D Additional Experiments

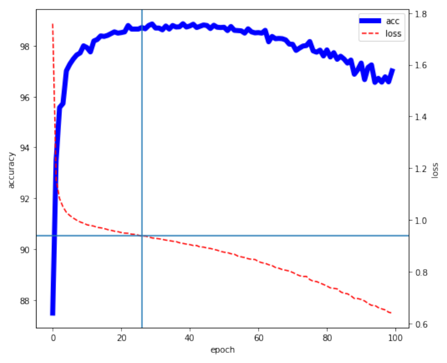
In this section, we give more experiment results on Gambler + Early Stopping. See Figure 7

E Definition of transition matrix

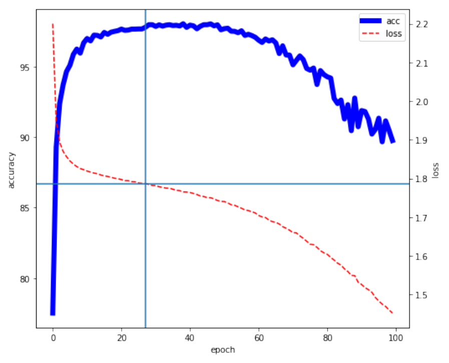
Transition matrix is used to replace some correct labels with wrong labels because the dataset used in our experiments such as MNIST and CIFAR are clean. There are two different types of noise transition matrix and explanation follows.

Symmetric Flipping: Each label has an uniform possibility to transform to one of the rest class. In most cases of manual label work, a correct label has equal possibility to be any other class label. n is the number of class and δ is the rate of label that will be modified as a wrong label

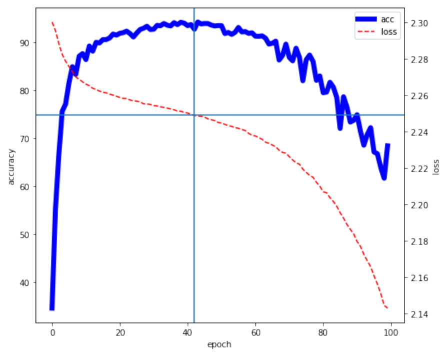
$$M := \begin{bmatrix} 1-\delta & \frac{\delta}{1-n} & \frac{\delta}{1-n} & \dots & \frac{\delta}{1-n} \\ \frac{\delta}{1-n} & 1-\delta & \frac{\delta}{1-n} & \dots & \frac{\delta}{1-n} \\ \frac{\delta}{1-n} & \frac{\delta}{1-n} & 1-\delta & \dots & \frac{\delta}{1-n} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{\delta}{1-n} & \frac{\delta}{1-n} & \frac{\delta}{1-n} & \dots & 1-\delta \end{bmatrix} \tag{14}$$



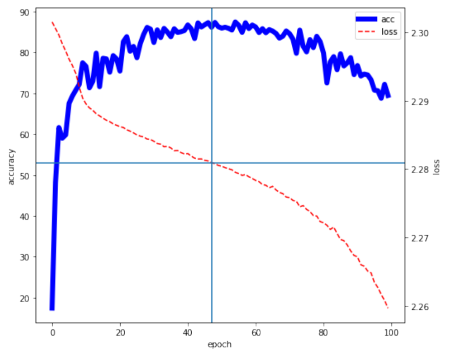
(a) MNIST: cr = 0.2



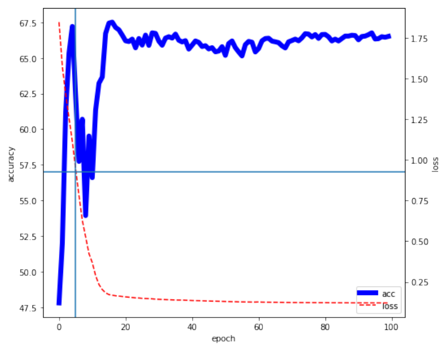
(b) cr = 0.5



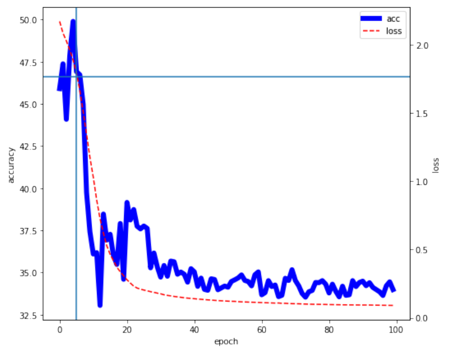
(c) cr = 0.8



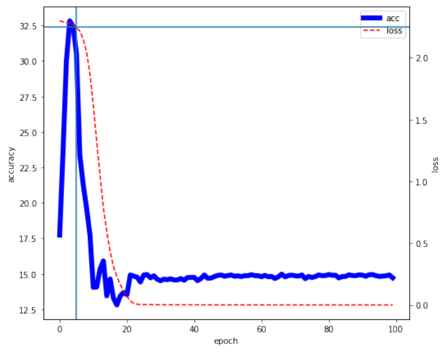
(d) cr = 0.85



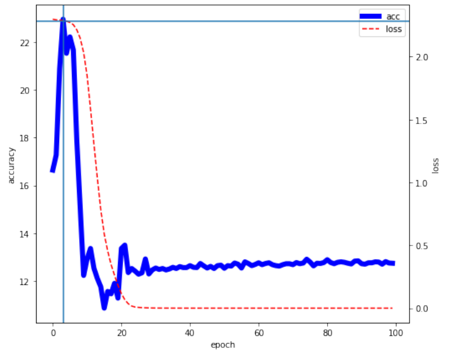
(e) CIFAR10: cr = 0.2



(f) cr = 0.5



(g) cr = 0.8



(h) cr = 0.85

Figure 7: Early stopping on MNIST (1st and 2nd row) and CIFAR10 (3rd and 4th row). cr refers to corruption rate. The horizontal line is the predicted early stopping point. We see that this point almost always corresponds to where the testing accuracy (vertical blue solid line) is at maximum.

Pair Flipping: Each class can only be corrupted as a specific class. For example, it's more likely to label 6 as 9 than any other class except 9. δ is the rate of label that will be modified as a wrong label:

$$M := \begin{bmatrix} 1-\delta & \delta & 0 & \dots & 0 \\ 0 & 1-\delta & \delta & \dots & 0 \\ 0 & 0 & 1-\delta & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \delta & 0 & 0 & \dots & 1-\delta \end{bmatrix} \tag{15}$$

F CNN ARCHITECTURE

In this section, we show the architectures that we used but did not describe in the main text.

Table 3: architecture of the neural network used in MNIST

CCNN on MNIST
28 × 28 Gray Image
20 × 5 conv, 1 ReLU
2 × 2 max-pool, stride 2
50 × 5 conv, 20 ReLU
2 × 2 max-pool, stride 2
dense 800 → 500
dense 500 → 11

Table 4: architecture of the neural network used in cifar10

CNN on cifar10
32 × 32 RGB Image
5 × 5 conv, 128 LReLU
5 × 5 conv, 128 LReLU
2 × 2 max-pool, stride 2
dense 128 → 128
dense 128 → 11

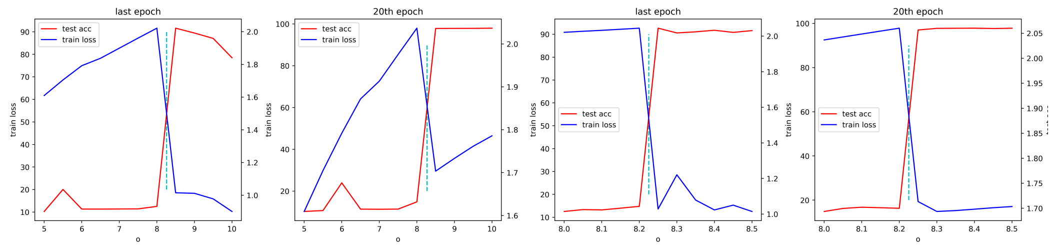
G EFFECT OF TUNING O AND CRITICAL BEHAVIOR

In this section, we give more experiments on the critical behavior discussed in 2.1. First see Figure 8.

In Figure 2, we show that lower value of o provides stronger robustness to label noise consistently; this raises the question about whether we can continue to lower o to its lower limit (= 1) to achieve the best performance or not - the answer is no, but the reason is very surprising. We empirically find that a first order phase transition exists when tuning o (such as transition of water to ice). In particular, we find that there exists a critical value o_{crit} such that, when $o > o_{\text{crit}}$, the robustness to label noise increases as we lower o; when $o < o_{\text{crit}}$, the learning proceeds badly: *the change in performance is discontinuous*. This is characteristic of a physical first order phase transition (Landau & Lifshitz, 2013). See Figure 3. In this setting, we have $r = 0.5$ for MNIST, and we pin down the o_{crit} to lie between 8.21 and 8.22; qualitatively different behaviors are found for the two different phase. See appendix for plots ranging from $o = 10.0$ to $o = 5.0$. A qualitatively similar behavior is observed using Adam[1] and on CIFAR10, and using other architectures. Clearly studying this critical behavior is very interesting but beyond the scope of this work. The above observation, however, suggests the following heuristic to tune hyperparameter o:

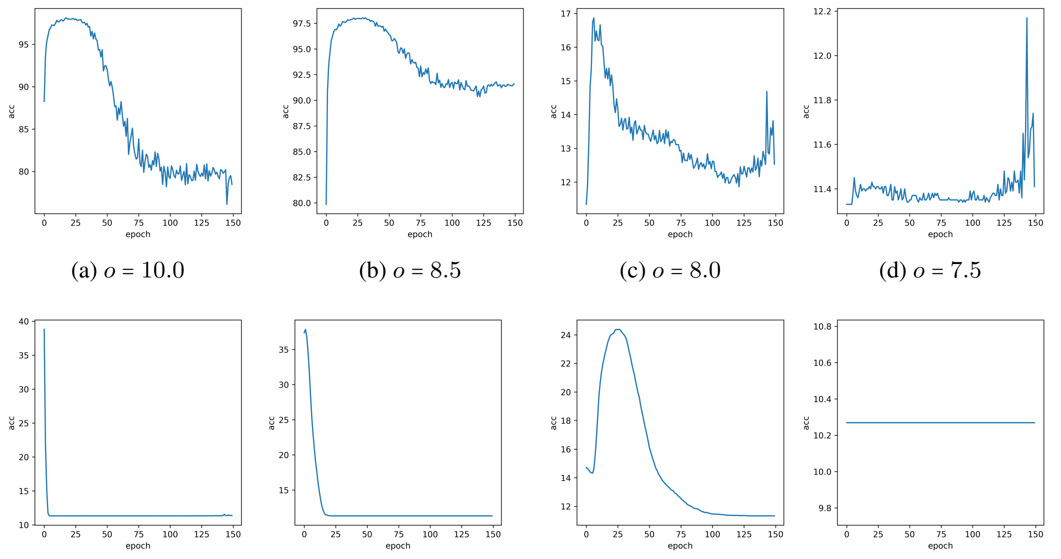
1. Start with o very close to M (for reference, for MNIST, $o \approx 9.5$ seems a good starting point; for CIFAR10, $o \approx 9.9$; CIFAR100: $o \approx 99.9$).

[1]Using Adam we can even pin down o_{crit} for this problem to lie between 8.2100 and 8.2101



(a) critical behavior around $o = 8.2$ at $epoch = 200$ (b) critical behavior around $o = 8.2$ at $epoch = 200$ (c) critical behavior around $o = 8.2$ at $epoch = 20$ (d) critical behavior around $o = 8.2$ at $epoch = 200$

Figure 8: critical behavior of the gambler's loss. We see that the learning almost do not happen at all for $o < o_{crit}$, while above o_{crit} the behavior is qualitatively similar. The optimal o can be tuned for using this phenomenon; however, more often one does not need to tune for o.



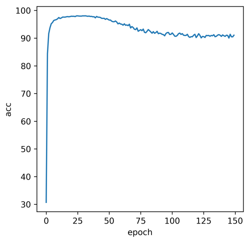
(a) $o = 10.0$ (b) $o = 8.5$ (c) $o = 8.0$ (d) $o = 7.5$

(e) $o = 7.0$ (f) $o = 6.5$ (g) $o = 6.0$ (h) $o = 5.0$

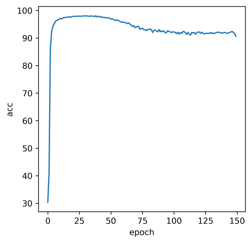
Figure 9: Gambler's loss is used alone without early stopping method. Initially, the performance gets better with the o decreasing. While the performance gets worse with the o decreasing when o is less than 8.5, which indicates there is a critical point for o locates in between 8.0 and 8.5. The most suitable o is just larger than the critical point

2. If the learning curve starts similarly as before, then decrease o until the learning stops abruptly, and use the previous o

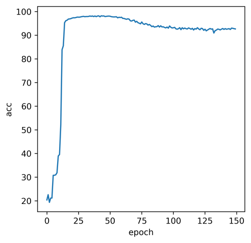
This procedure for tuning o does not rely on a validation set at all. Also, as show in in Figure 2 for $o = 8.3, 8.5$, the performance is very similar when o is close to o_{crit}, there is no need to be infinitesimally close to the critical value. Unless otherwise noted, we tune our hyperparameter this way in the experiment section. Moreover, One can prove that $o = M$ will always work at least as good as the baseline, and so o_{cr} should always $< M$.



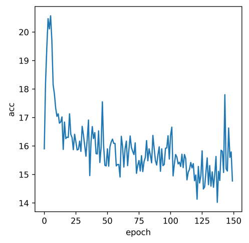
(a) o = 8.35



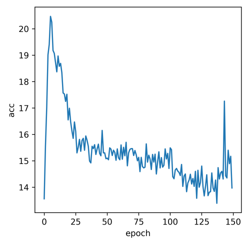
(b) o = 8.3



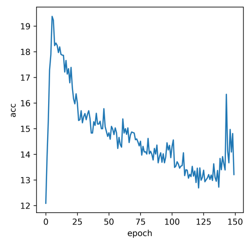
(c) o = 8.25



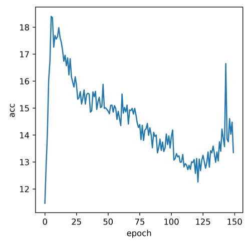
(d) o = 8.2



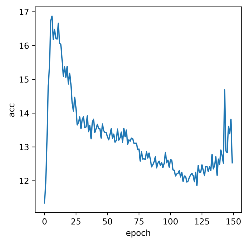
(e) o = 8.15



(f) o = 8.1



(g) o = 8.05



(h) o = 8.0

Figure 10: Under higher resolution on hyperparameter o, it's for sure that the performance will get better when o gets closer to critical point from right side on number axis. While the performance will dramatically get terrible when o is just lower than the critical points. What's more, the critical point is in the range of $[8.2, 8.25]$

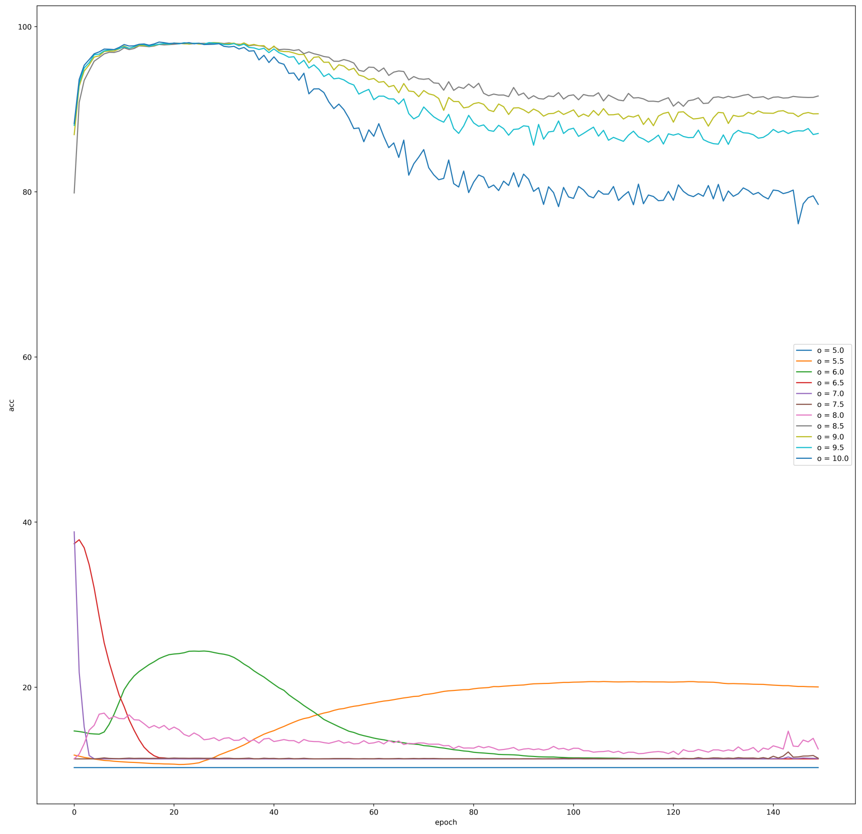


Figure 11: put all curve of Figure 8 in one Figure