Pumpout: A Meta Approach to Robust Deep Learning with Noisy Labels

Bo Han\textsuperscript{1,2}, Gang Niu\textsuperscript{1}, Jiangchao Yao\textsuperscript{2}, Xingrui Yu\textsuperscript{2}, Miao Xu\textsuperscript{1}, Ivor W. Tsang\textsuperscript{2}, and Masashi Sugiyama\textsuperscript{1,3}

\textsuperscript{1}Center for Advanced Intelligence Project, RIKEN, Japan
\textsuperscript{2}Center for Artificial Intelligence, University of Technology Sydney, Australia
\textsuperscript{3}Graduate School of Frontier Sciences, University of Tokyo, Japan

\{bo.han, gang.niu, miao.xu\}@riken.jp
\{jiangchao.yao, xingrui.yu\}@student.uts.edu.au
ivor.tsang@uts.edu.au, sugi@k.u-tokyo.ac.jp

Abstract. Recent studies reveal that deep neural networks gradually memorize individual data while fitting distributions of data. Hence, when facing noisy labels, all existing methods inevitably suffer from generalization degeneration and have to be early stopped. In this paper, we propose Pumpout as a meta approach to learning with noisy labels and an alternative to early stopping. Pumpout comes from sample selection and goes beyond: in every mini-batch, it uses gradient descent on good data, while it uses scaled gradient ascent on bad data rather than drops those data, where the goodness and badness are w.r.t. a base learning method. It is advantageous over early stopping, since it can continue to fit distributions of data and it has the ability of actively forgetting individual data that is memorized by mistakes. We demonstrate via experiments that Pumpout robustifies two representative base learning methods, and the performance boost is often significant.

1 Introduction

Labels of data in industry are heavily noisy, and their label generation processes are usually agnostic \cite{28,10}. Therefore, learning from these data is quite demanding. Essentially, noisy labels of such data are corrupted from ground-truth labels \cite{20}, which degenerates the robustness of learned models. It is noted that industrial-level data is frequently emerging in our daily life, such as social-network data \cite{2}, E-commerce data \cite{28} and crowdsourcing data \cite{27}.

Due to the large data volume, industrial-level data can be handled by deep neural networks \cite{28}. Thus, the key issue is how to train deep neural networks robustly on noisy labels of such data, since deep neural networks have the high capacity to fit noisy labels eventually \cite{29}. To handle noisy labels, one base learning approach focuses on estimating the label transition matrix \cite{510}, which models the label corruption process. For instance, \cite{22} first leveraged a two-step

\textsuperscript{1} Preprint. Work in progress.
solution to estimate the label transition matrix. Based on this estimated matrix, they conducted backward loss correction, which is used for training deep neural networks robustly.

Nonetheless, the label transition matrix is hard to be estimated accurately, especially when the number of classes is large. Motivated by the memorization in deep neural networks [1], another base learning approach becomes emerging. This approach focuses on training only on selected instances [10][24], which does not require any prior assumptions on noisy labels. Specifically, deep learning models are known to memorize easy instances first, then gradually adapt to hard instances when training epochs become large [1]. Therefore, in the first few iterations, we may train deep neural networks on the whole dataset, and let them enough learn clean instances in the noisy dataset. Then we employ the small-loss trick [10][7], which tries to perform the training selectively on small-loss instances.

However, when noisy labels indeed exist, no matter estimating the label transition matrix or training on small-loss instances, deep networks inevitably memorize some noisy labels, which will lead to the poor generalization performance [29][1]. In this paper, on the top of base learning approaches, we design a meta algorithm called Pumpout, which can enhance base learning approaches to mitigate the issue of memorizing noisy labels. The main idea of Pumpout is to actively squeeze out the negative effects of noisy labels from the model being trained, instead of passively forgetting these effects by further training. Specifically, on good data, Pumpout conducts stochastic gradient descent typically; while on bad data, Pumpout conducts scaled stochastic gradient ascent, instead of stopping gradient computation as usual. This aggressive policy can erase the negative effects of noisy labels actively.

We leverage meta algorithm Pumpout to robustify two representative base learning approaches: MentorNet [10] and Backward Correction [22][26]. Specifically, MentorNet is noise-model-free, while Backward Correction is noise-model-based. They are good at different learning tasks. We conducted experiments on benchmark vision and text datasets, namely simulated noisy MNIST, CIFAR-10 and NEWS datasets. Empirical results demonstrated that, under extremely noisy labels and low-level noisy labels, Pumpout enhances the robustness of two base learning approaches, and the performance boost is often significant.

2 Pumpout Meets Noisy Supervision

Idea of meta algorithm. The original idea of Pumpout is to actively squeeze out the negative effects of noisy labels from the training model, instead of passively forgetting these effects. Intuitively, take “summing up a list of 100 numbers” as a motivating example. Suppose after we finish all additions, we are told by our boss that 10 numbers should be excluded—what could we do? Of course, we can clear the summation result and then add again the 90 included numbers. However, we can also keep the summation result and just subtract the 10 excluded numbers from it. The latter is obviously a better idea than the former. In this
Pumpout: A Meta Approach to Robust Deep Learning with Noisy Labels

intuitive example, the summation result is our model, the additions make our model memorize included numbers, and the subtractions make our model pump excluded numbers out.

In the design of meta algorithm, we should consider how our meta algorithm can simultaneously robustify multiple base learning approaches to combat with noisy labels. Commonly, there are three base learning directions: training on selected instances [10], estimating the label transition matrix [22] and designing regularization [19,16]. For this purpose, we generalize noisy labels into “not-fitting” (or bad) labels, and generalize clean labels into “fitting” (or good) labels (details of the fitting condition will be discussed in Q1 below). In the high level, the meta algorithm Pumpout is to train deep neural networks by stochastic gradient descent on “fitting” labels, and train deep neural networks by scaled stochastic gradient ascent on “not-fitting” labels. In the low level, the proposed Algorithm 1 is named Pumpout.

Realization of meta algorithm. To realize Pumpout, we maintain deep neural network $f$ (with parameter $w_f$). When a single point $\{x_i, y_i\}$ is sequentially selected from a noisy mini-batch $\mathcal{D}$ (step 5), we first calculate the temporary gradient $g_t$ (step 6) where $\ell(x_i, y_i; w_f)$ is a vector of the negative logarithm of the softmax layer. Then, we check whether $\{x_i, y_i\}$ is fitting the discriminative condition or not. If yes, we positively accumulate gradient (step 7); otherwise, we reversely accumulate scaled ($\gamma$) gradient (step 8), which erases the negative effects of “not-fitting” labels. These “not-fitting” labels hinder us to train a robust model. After finishing gradient accumulation in each batch, we average the accumulated gradients $G_a$ (step 9), and update parameter $w_f$ by stochastic optimization (step 10).

Note that, stochastic gradient descent is realized by positive gradient accumulation (step 7) with stochastic optimization (step 10). Scaled stochastic gradient ascent is realized by reversely scaled gradient accumulation (step 8) with stochastic optimization (step 10). The abstract algorithm arises three important questions as follows.

Three important questions.

Q1. What is the fitting condition?
Q2. Why do we need gradient ascent on non-fitting data, in addition to gradient descent on fitting data?
Q3. Why do we need to scale the stochastic gradient ascent on non-fitting data?

To answer the first question, we need to emphasize a view that orthogonal base approaches require different fitting conditions. Intuitively, if a single point $\{x_i, y_i\}$ satisfies a discriminative fitting condition, it means that our training model will regard this data point as useful knowledge, and fitting on this point will benefit training the robust model. Conversely, if a single point $\{x_i, y_i\}$ does not satisfy the discriminative fitting condition, it means that, our training model will regard this data point as useless knowledge, and want to erase the negative
Algorithm 1 Meta Algorithm Pumpout.

1: **Input** deep network $f$ with parameter $w_f$, training set $\mathcal{D}$, batch size $B$, learning rate $\eta$, maximum epoch $T_{\text{max}}$, hyper parameter $0 \leq \gamma \leq 1$;

for $t = 1, 2, \ldots, T_{\text{max}}$ do

2: Shuffle $\mathcal{D}$ into $\frac{|\mathcal{D}|}{B}$ mini-batches; //noisy dataset $\mathcal{D}$

for $n = 1, \ldots, \frac{|\mathcal{D}|}{B}$ do

3. Set $G_a = 0$; //gradient accumulator

4: Draw $n$-th mini-batch $\bar{\mathcal{D}}$ from $\mathcal{D}$;

for $i = 1, \ldots, B$ do

5: Select $\{x_i, y_i\}$ from $\bar{\mathcal{D}}$ sequentially;

6: Set $g_t = \nabla_{w_f} \{1^\top f(x_i, y_i; w_f)\}$; //temporary gradient

if $\{x_i, y_i\}$ is fitting then

7: Set $G_a = G_a + g_t$; //positive gradient accumulation

else

8: Set $G_a = G_a - \gamma g_t$; //reversely scaled gradient accumulation

end

end

9: Average $q_a = G_a / B$;

10: Update $w_f = w_f - \eta q_a$; // This step corresponds to SGD, which can be replaced with any stochastic optimization (e.g., ADAM [11]).

end

end

11: **Output** $w_f$.



effects of this point actively. To instantiate the fitting condition, we provide two concrete cases in Algorithm 2 and Algorithm 3, respectively.

The above answer motivates our second question: why cannot we only conduct stochastic gradient descent on fitting data points (mainly by step 7). In other words, can we remove scaled stochastic gradient ascent (mainly by step 8) in Algorithm 1? In this case (removing step 8), our algorithm degenerates to training only on selected instances. However, once some of the selected instances are found to be false positives\(^2\), our training model will fit on them, and thus the negative effects will inevitably occur (i.e., degrading the generalization (test accuracy)). Instead of passively forgetting these negative effects (i.e., further training over many epochs), we hope to actively squeeze out the negative effects from the training model by using scaled stochastic gradient ascent (mainly by step 8).

Lastly, the third question closely connects with the second one. Namely, why do we need *scaled* instead of *ordinary* stochastic gradient ascent? The answer can be intuitively explained. Assume that we view stochastic gradient ascent as a correction to “not-fitting” labels, and view $0 \leq \gamma \leq 1$ as a scale parameter. When $\gamma = 1$, our Pumpout will squeeze out the negative effects with the full fast rate; while when $\gamma = 0$, our Pumpout will not squeeze out any negative effects. Both cases are not optimal, and we empirically find that the best performance

\(^2\)https://en.wikipedia.org/wiki/False_positives_and_false_negatives
Pumpout: A Meta Approach to Robust Deep Learning with Noisy Labels

is usually chosen when $0 < \gamma < 1$ by using the validation set (Section 4). For the first case, the fast squeezing rate will negatively affect the convergence of our algorithm. For the second case, no squeezing rate will inevitably let deep neural networks memorize some “not-fitting” labels, which lowers their generalization [29][21].

3 Pumpout Robustifies State-of-the-Art Base Learning Approaches

In this section, we employ the meta algorithm Pumpout to robustify two representative base learning approaches: MentorNet and Backward Correction. First, we briefly introduce the background of MentorNet and Backward Correction. Then, by using Pumpout, we propose enhanced MentorNet (PumpoutSL) and enhanced Backward Correction (PumpoutBC). Lastly, we explain the relation between MentorNet (resp. Backward Correction) and PumpoutSL (resp. PumpoutBC).

3.1 Pumpout Robustifies MentorNet

MentorNet. To handle noisy labels, an emerging base learning direction focuses on training only on selected instances [10][24][7], which is free of estimating the label transition matrix, and also free of the class-conditional noise assumption [20]. These works try to select clean instances out of the noisy ones, and then use them to update the network. Among those works, a representative method is MentorNet [10], which employs the small-loss trick. Specifically, MentorNet pre-trains an extra network, and then uses the extra network for selecting small-loss instances as clean instances to guide the training. However, the idea of MentorNet is similar to the self-training approach [3], thus MentorNet inherits the same drawback of accumulated error.

PumpoutSL. Algorithm 2 represents the enhanced MentorNet using Pumpout approach (denoted as PumpoutSL), where MentorNet uses the small-loss trick. Specifically, we maintain deep neural network $f$ (with parameter $w_f$). When a mini-batch $\bar{D}$ is formed (step 4), we first let $f$ select a small proportion of instances $\bar{D}_s$ in this mini-batch that have small training losses (step 5). The number of selected instances is controlled by $R(t)$, and $f$ only samples $R(t)$ percentage of instances out of the mini-batch.

Further when a single point $\{x_i, y_i\}$ is sequentially selected from the mini-batch $\mathcal{D}$ (step 6), we first compute the temporary gradient $g_t$ at this point (step 7). If this point belongs to small-loss instances, namely $\{x_i, y_i\} \in \mathcal{D}_s$, we accumulate gradient $G_s$ by positive gradient (step 8); otherwise, we accumulate gradient $G_a$ by reversely scaled gradient (step 9), and this step erases the negative effects of big-loss instances. Lastly, we average the accumulated gradient (step 10) and update parameter $w_f$ by stochastic optimization (step 11). The update of $R(t)$ (step 12) follows [7], in which extensive discussion has been conducted.
Algorithm 2 Pumpout\textsubscript{SL}. The fitting condition is whether a point belongs to small-loss instances $\bar{D}_s$.

1: \textbf{Input} deep network $f$ with parameter $w_f$, training set $D$, batch size $B$, learning rate $\eta$, estimated noise rate $\tau$, maximum epoch $T_{\text{max}}$, hyper parameter $0 \leq \gamma \leq 1$; 
for $t = 1, 2, \ldots, T_{\text{max}}$ do 
\hspace{1em} 2: Shuffle $D$ into $|D|$ mini-batches; //noisy dataset $D$ 
\hspace{1em} for $n = 1, \ldots, N$ do 
\hspace{2em} 3. Set $G_a = 0$; //gradient accumulator 
\hspace{2em} 4: Draw $n$-th mini-batch $\bar{D}$ from $D$; 
\hspace{2em} 5: Draw $D' = \arg\min_{D'||D'| \geq R(t)} \ell(D'; w_f)$; //sample $R(t)\%$ small-loss instances 
\hspace{2em} for $i = 1, \ldots, B$ do 
\hspace{3em} 6: Select $\{x_i, y_i\}$ from $\bar{D}$ sequentially; 
\hspace{3em} 7: Set $g_i = \nabla_{w_f} \ell(x_i, y_i; w_f)$; //temporary gradient 
\hspace{3em} if $\{x_i, y_i\} \in D_s$ then 
\hspace{4em} 8: Set $G_a = G_a + g_i$; //positive gradient accumulation 
\hspace{3em} else 
\hspace{4em} 9: Set $G_a = G_a - \gamma g_i$; //reversely scaled gradient accumulation 
\hspace{2em} end 
\hspace{2em} 10: Average $g_n = G_a / B$; 
\hspace{2em} 11: Update $w_f = w_f - \eta g_n$; // This step corresponds to SGD. 
\hspace{2em} end 
\hspace{1em} 12: Update $R(t) = 1 - \min\left\{\frac{1}{\pi}, \tau\right\}$; 
end 
13: \textbf{Output} $w_f$.

Relation between MentorNet and Pumpout\textsubscript{SL}. If we remove step 9 in Algorithm 2, Pumpout\textsubscript{SL} algorithm will be reduced to the core version of MentorNet, namely self-paced MentorNet. It means that Pumpout\textsubscript{SL} algorithm is more aggressive than MentorNet in essence. Namely, Pumpout\textsubscript{SL} conducts not only stochastic gradient descent on small-loss instances (like MentorNet), but also scaled stochastic gradient ascent on big-loss instances.

3.2 Pumpout Robustifies Backward Correction

Backward Correction and its non-negative version. To handle noisy labels, another popular base learning direction focuses on estimating the label transition matrix [5,22,16]. Among those works, a representative method is Backward Correction. Specifically, [22] leveraged a two-step solution to estimate the label transition matrix heuristically. Then they employed the estimated matrix to correct the original loss, and robustly train a deep neural network based on the new loss function.

Theorem 1. (Backward Correction, Theorem 1 in [22]) Suppose that the label transition matrix $T$ is non-singular, where $T_{ij} = \Pr(\hat{y} = j | y = i)$ given that

\begin{equation}
\end{equation}
noisy label $\tilde{y} = j$ is flipped from clean label $y = i$. Given loss $\ell$ and network parameter $w_f$, Backward Correction is defined as

$$\ell^-(x, y; w_f) = T^{-1}\ell(x, y; w_f). \quad (1)$$

Then, corrected loss $\ell^-(x, y; w_f)$ is unbiased, namely,

$$E_{\tilde{y}\mid x}\ell^-(x, y; w_f) = E_{y\mid x}\ell(x, y; w_f), \forall x. \quad (2)$$

Remark 1. Backward Correction operates on the loss vector directly. It is unbiased. LHS of Eq. (2) draws from noisy labels, and RHS of Eq. (2) draws from clean labels. Note that the corrected loss is differentiable, but not always non-negative [26].

If the model being trained is flexible, such as a deep neural network, the backward loss correction will lead to negative risks, and the hazardous aspect is to yield an over-fit issue. Previous work demonstrates that a negative but lower bounded risk function can still result in terrible over-fitting issue in PU learning [12]. Following a similar motivation, we conduct a non-negative correction again based on the backward-corrected loss, since the risk should always be greater than 0 or equal to.

**Theorem 2.** (Non-negative Backward Correction) Suppose that the label transition matrix $T$ is non-singular, where $T_{ij} = \Pr(\tilde{y} = j \mid y = i)$ given that noisy label $\tilde{y} = j$ is flipped from clean label $y = i$. Given loss $\ell$ and network parameter $w_f$, Non-negative Backward Correction is defined as

$$\ell_m^-(x, y; w_f) = \max\{0, 1^T T^{-1}\ell(x, y; w_f)\}, \quad (3)$$

where $1_{k\times1}$. Then, the corrected loss $\ell_m^-(x, y; w_f)$ is non-negative.

Remark 2. $\ell_m^-(x, y; w_f)$ is a non-negative scalar. Our key claim is to overcome the over-fit issue by non-negative correction.

However, the above non-negative correction is passive, since max operator means stopping gradient computation on negative-risk instances. This correction may not achieve the optimal performance. Namely, when $1^T T^{-1}\ell(x, y; w_f) \geq 0$, we conduct stochastic gradient descent; otherwise, we do not perform the operation of stochastic gradient. To propose an aggressive non-negative correction, we reverse the gradient computation at negative-risk instances. Specifically, we use the Pumpout approach to improve Non-negative Backward Correction. Namely, when $1^T T^{-1}\ell(x, y; w_f) \geq 0$, we conduct stochastic gradient descent typically; when $1^T T^{-1}\ell(x, y; w_f) \leq 0$, we conduct scaled stochastic gradient ascent. This brings our Algorithm 3.
Algorithm 3 Pumpout$_{BC}$. The fitting condition is whether a point satisfies $1^\top T^{-1}(x_i, y_i; w_f) \geq 0$.

1: **Input** deep network $f$ with parameter $w_f$, training set $D$, batch size $B$, learning rate $\eta$, maximum epoch $T_{\text{max}}$, hyper parameter $0 \leq \gamma \leq 1$;
for $t = 1, 2, \ldots, T_{\text{max}}$ do
2: Shuffle $D$ into $\left\lfloor \frac{|D|}{B} \right\rfloor$ mini-batches; //noisy dataset $D$
for $n = 1, \ldots, \left\lfloor \frac{|D|}{B} \right\rfloor$ do
3: Set $G_a = 0$; //gradient accumulator
4: Draw $n$-th mini-batch $\bar{D}$ from $D$;
for $i = 1, \ldots, B$ do
5: Select $\{x_i, y_i\}$ from $\bar{D}$ sequentially;
6: Set $g_t = \nabla_{w_f} 1^\top T^{-1}(x_i, y_i; w_f)$; //temporary gradient
if $1^\top T^{-1}(x_i, y_i; w_f) \geq 0$ then
7: Set $G_a = G_a + g_t$; //positive gradient accumulation
else
8: Set $G_a = G_a - \gamma g_t$; //reversely scaled gradient accumulation
end
end
9: Average $g_a = G_a / B$;
10: Update $w_f = w_f - \eta g_a$; // This step corresponds to SGD.
end
end
11: **Output** $w_f$.

*Pumpout$_{BC}$. Algorithm 3 represents the enhanced Backward Correction using the Pumpout approach (denoted as Pumpout$_{BC}$), where Backward Correction is defined in Theorem 1. If the model being trained is flexible (i.e., a deep neural network), Backward Correction will lead to negative risks [22], which subsequently yields an over-fit issue. To mitigate this issue, we maintain deep neural network $f$ (with parameter $w_f$). When a single point $\{x_i, y_i\}$ is sequentially selected from the mini-batch $\bar{D}$ (step 5), we first compute the temporary gradient $g_t$ at this point (step 6). If Backward Correction produces a positive risk at this point, namely $1^\top T^{-1}(x_i, y_i; w_f) \geq 0$ (definitions of $T$ and $\ell$ are in Theorem 1), we accumulate gradient $G_a$ by positive gradient (step 7); otherwise, we accumulate gradient $G_a$ by reversely scaled gradient (step 8), and this step erases the negative effects of negative-risk instances. Lastly, we average the accumulated gradient (step 9) and update parameter $w_f$ by stochastic optimization (step 10).

Relation between Non-negative Backward Correction and Pumpout$_{BC}$. If we remove line 8 in Algorithm 3, Pumpout$_{BC}$ algorithm will be reduced to Non-negative Backward Correction. It means Pumpout$_{BC}$ algorithm is an aggressive version of Non-negative Backward Correction. Namely, Pumpout$_{BC}$ conducts not only stochastic gradient descent on nonnegative-risk instances, but also scaled stochastic gradient ascent on negative-risk instances to erase their negative effects.
4 Experiments

Datasets. We verify the effectiveness of our Pumpout approach on three benchmark datasets, including two vision datasets and one text dataset. MNIST, CIFAR-10 and NEWS are used here (Table 1), as these data sets are popularly used for evaluation of noisy labels in the literature [23,22,14].

|          | # of training | # of testing | size of image/text |
|----------|---------------|--------------|--------------------|
| MNIST    | 60,000        | 10,000       | 28×28              |
| CIFAR-10 | 50,000        | 10,000       | 32×32              |
| NEWS     | 11,314        | 7,532        | 300-D              |

Since all datasets are clean, following [23,22], we need to corrupt these datasets manually by the label transition matrix $T$, where $T_{ij} = \Pr(\tilde{y} = j | y = i)$ given that noisy $\tilde{y}$ is flipped from clean $y$. Assume that the matrix $T$ has two representative structures: (1) Pair flipping [6]: a real-world application is the fine-grained classification, where you may make mistake only within very similar classes in the adjunct positions; (2) Symmetry flipping [25]. Their precise definition is in Appendix A.

This paper first verifies whether Pumpout can significantly improve the robustness of representative methods on extremely noisy supervision, the noise rate $\tau$ is chosen from {0.45, 0.5}. Intuitively, this means almost half of the instances have noisy labels. Note that, the noise rate $> 50\%$ for pair flipping means over half of the training data have wrong labels that cannot be learned without additional assumptions. In addition to extremely noisy settings, we also verify whether Pumpout can significantly improve the robustness of representative methods on low-level noisy supervision, where $\tau$ is set to 0.2. Note that pair case is much harder than symmetry case (Appendix A).

Baselines. To verify the efficacy of Pumpout, we select two representative base learning approaches to enhance. Note that, Pumpout can be viewed as a “if-else” algorithm. To fairly verify this “if-else” algorithm, we should compare it with “if” algorithm and “none” algorithm.

The first set (SET1) comparison is to check whether Pumpout can improve the robustness of MentorNet.

- Standard deep network that directly trains on the noisy set (denoted as “Standard”) is “none” algorithm.
- MentorNet [10] is “if” algorithm.
- Pumpout SL (Algorithm 2) is “if-else” algorithm;

The second set (SET2) is to check whether Pumpout can improve the robustness of Backward Correction.
Fig. 1. Results of Pumpout SL and MentorNet on MNIST dataset. Top: test accuracy vs. number of epochs; bottom: label precision vs. number of epochs. In Figure 1(c), the accuracy degeneration of Standard is small (about 0.1), which is similar to [1].

- Backward Correction [22] (denoted as “BC”, Theorem 1) is “none” algorithm.
- Non-negative backward correction (denoted as “nBC”, Theorem 2) is “if” algorithm.
- Pumpout BC (Algorithm 3) is “if-else” algorithm.

Besides, the choice of two baselines is to justify whether Pumpout can robustify state-of-the-art base learning approaches. The readers are encouraged to robustify other base learning methods, such as Bootstrap [23], S-model [5] and Co-teaching [7] by using Pumpout.

We implement all methods with default parameters by PyTorch, and conduct all the experiments on a NVIDIA K80 GPU. Standard CNN is used with Leaky ReLU (LReLU) activation function [18]; ResNet is used with ReLU activation function; and MLP is used with Softsign activation function [4]. The detailed architectures are in Appendix B. Namely, we used the 9-layer CNN [19,13] with dropout and batch-normalization for MNIST, ResNet-32 [8] with batch-normalization for CIFAR-10, and 3-layer MLP [12] with batch-normalization for NEWS, since the network structures we used here are standard test bed for weakly-supervised learning. For all datasets, Adam optimizer (momentum=0.9) with an initial learning rate of 0.001, the batch size is set to 128 and runs for 200 epoch. Note that, the focus of our paper is to explore the efficacy of Pumpout. Therefore, we use Adam optimizer in all experiments for fair comparison without using data augmentation trick [30,17].
Experimental setup. The ratio of small-loss instances $R(t)$ is an important parameter for Pumpout$_{SL}$ and MentorNet. Here, we assume the noise rate $\tau$ is known and set $R(t) = 1 - \tau \cdot \min \left( t/T_k, 1 \right)$ with $T_k = 10$. If $\tau$ is not known in advanced, $\tau$ can be inferred using validation sets [15]. The choices of $R(t)$ and $\tau$ follows [7]. Note that $R(t)$ only depends on the memorization effects of deep networks but not any specific datasets. The scale of gradient ascent $\gamma$ is an important parameter for Pumpout$_{SL}$ and Pumpout$_{BC}$, where $0 \leq \gamma \leq 1$. The choices of $\gamma$ follows [12], and $\gamma$ is chosen among $\{0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ via a validation set.

This paper provides two enhanced approaches to train deep neural networks robustly under noisy labels. Thus, our goal is to classify the clean instances as accurately as possible, and the measurement for both SET1 and SET2 is the test accuracy, i.e., \text{test accuracy} = \left( \frac{\text{# of correct predictions}}{\text{# of test dataset}} \right). Besides, for SET1, we also use the label precision in each mini-batch, i.e., \text{label precision} = \left( \frac{\text{# of clean labels}}{\text{# of all selected labels}} \right). Specifically, we sample $R(t)$ of small-loss instances in each mini-batch, and then calculate the ratio of clean labels in the small-loss instances. Intuitively, higher label precision means less noisy instances in the mini-batch after sample selection; and the algorithm with higher label precision is also more robust to the label noise. For SET2, we cannot report the label precision, since the family of Backward Correction does not sample instances. All experiments are repeated five times. In each figure, the error bar for standard deviation is highlighted as a shade.
Before delving into Section 4.1 and 4.2, there are two important points to be emphasized. First, the memorization effects of deep networks \cite{1} means that standard deep networks will fit clean instances first, then overfit noisy instances gradually. These effects will inevitably lower the generalization performance (i.e., test accuracy). Second, Pumpout\textsubscript{BC} may not suffer from or greatly alleviate the accumulated error in MentorNet, since Pumpout\textsubscript{SL} actively squeezes out the negative effects of noisy labels from the training model, instead of passively forgetting these effects. Due to the limited space, we move empirical results on NEWS in Appendix C.

4.1 Results of Pumpout\textsubscript{SL} and MentorNet

\textit{MNIST.} In Figure 1, we show test accuracy (top) and label precision (bottom) vs. number of epochs on MNIST dataset. In all three plots, we can clearly see the memorization effects of deep networks \cite{1}, i.e., test accuracy of Standard first reaches a very high level and then gradually decreases. Thus, a good robust training method should stop or alleviate the decreasing process. On this point, our Pumpout\textsubscript{SL} almost stops the decreasing process in the easier Symmetric-50\% and Symmetric-20\% cases. Meanwhile, compared to MentorNet, our Pumpout\textsubscript{SL} alleviates the decreasing process in the hardest Pair-45\% case. Thus, Pumpout\textsubscript{SL} consistently achieves the higher accuracy over MentorNet.

To explain such good performance, we plot label precision (bottom). Compared to Standard, we can clearly see that both Pumpout\textsubscript{SL} and MentorNet
can successfully pick clean instances out. However, our Pumpout$_{SL}$ achieves the higher label precision on not only the easier Symmetric-50% and Symmetric-20% cases, but also the hardest Pair-45% case. This shows our approach is better at finding clean instances due to the usage of scaled stochastic gradient ascent.

**CIFAR-10.** Figure 2 shows test accuracy and label precision vs. number of epochs on CIFAR-10 dataset. Again, on test accuracy, we can see Pumpout$_{SL}$ strongly stops the memorization effects of deep networks. More importantly, on the easier Symmetric-50% and Symmetric-20% cases, it works better and better along with the training epochs. On label precision, while Standard fails to find clean instances, both Pumpout$_{SL}$ and MentorNet can do this. However, due to the usage of scaled stochastic gradient ascent, Pumpout$_{SL}$ is stronger and find more clean instances.

4.2 Results of Pumpout$_{BC}$ and nnBC

**MNIST.** Top of Figure 3 shows test accuracy vs. number of epochs on MNIST dataset. In the hardest Pair-45% case, both our Pumpout$_{BC}$ and nnBC can fully stop the performance decreasing process resulted from the memorization effects of deep networks. However, Pumpout$_{BC}$ significantly outperforms nnBC due to the usage of scaled stochastic gradient ascent. Meanwhile, in the easier Symmetric-50% and Symmetric-20% cases, our Pumpout$_{BC}$ works better and better along with the training epochs though it fluctuates. Besides, Pumpout$_{BC}$ finally achieves the higher accuracy over both BC and nnBC.

**CIFAR-10.** Bottom of Figure 3 shows test accuracy vs. number of epochs on CIFAR-10 dataset. In the hardest Pair-45% case and the easiest Symmetry-20% case, our Pumpout$_{BC}$ overcomes the decreasing issue and works better and better along with the training epochs though it fluctuates slightly. Specifically, in both cases, our Pumpout$_{BC}$ finally achieves the higher accuracy over both BC and nnBC. Meanwhile, in the Symmetric-50% case, our Pumpout$_{BC}$ becomes comparable with other methods.

4.3 Reflection of results

To sum up, Standard and BC can be concluded as “none” algorithm free of extra operations. MentorNet and nnBC can be viewed as “if” algorithm. Namely, if the fitting condition is satisfied, they conduct stochastic gradient descent. Moving further, Pumpout$_{SL}$ and Pumpout$_{BC}$ can be regarded as “if-else” algorithm. Namely, if the fitting condition is satisfied, they conduct stochastic gradient descent; else, they conduct *scaled stochastic gradient ascent*, instead of stopping gradient computation as usual. Due to the usage of scaled stochastic gradient ascent, meta algorithm Pumpout can robustify base learning approaches, which has been demonstrated in empirical results.
5 Conclusion

This paper presents a meta algorithm called Pumpout, which significantly improves the robustness of state-of-the-art base learning methods under noisy labels. Our key idea is to squeeze out the negative effects of noisy labels actively from the model being trained, instead of passively forgetting these effects. The realization of Pumpout is to train deep neural networks by stochastic gradient descent on “fitting” labels; while train deep neural networks by scaled stochastic gradient ascent on “not-fitting” labels. To demonstrate the efficacy of Pumpout, based on MentorNet and Backward Correction, we design two enhanced versions called Pumpout\textsubscript{SL} and Pumpout\textsubscript{BC}. The experimental results show that, both enhanced approaches can train deep models more robustly over original ones. In future, we can leverage Pumpout to train deep models for another weak supervision, e.g., complementary labels \cite{Ishida17}. Besides, we should investigate theoretical guarantees for Pumpout, and adapt our Pumpout to harder cases, e.g., Clothing1M \cite{Han18}.

Acknowledgments. MS was supported by the International Research Center for Neurointelligence (WPI-IRCN) at The University of Tokyo Institutes for Advanced Study. IWT was supported by ARC FT130100746, DP180100106 and LP150100671. Bo Han would like to thank the financial support from Center for Advanced Intelligence Project, RIKEN.

References

1. Arpit, D. and Jastrzebski, S. and Ballas, N. and Krueger, D. and Bengio, E. and Kanwal, M.S. and Maharaj, T. and Fischer, A. and Courville, A. and Bengio, Y.: A closer look at memorization in deep networks. In: ICML. (2017)
2. Cha, Y. and Cho, J.: Social-network analysis using topic models. In: SIGIR. (2012)
3. Chapelle, Olivier and Scholkopf, Bernhard and Zien, Alexander: Semi-supervised learning. IEEE TNN. (2009)
4. Glorot, X. and Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. (2010)
5. Goldberger, J. and Ben-Reuven, E.: Training deep neural-networks using a noise adaptation layer. In: ICLR. (2017)
6. Han, B. and Yao, J. and Niu, G. and Zhou, M. and Tsang, I. and Zhang, Y. and Sugiyama, M.: Learning with noisy labels. In: NeurIPS. (2018)
7. Han, B. and Yao, J. and Niu, G. and Zhou, M. and Tsang, I. and Zhang, Y. and Sugiyama, M.: Co-teaching: Robust training of deep neural networks with extremely noisy labels. In: NeurIPS. (2018)
8. He, K. and Zhang, X. and Ren, S. and Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
9. Ishida, T. and Niu, G. and Hu, W. and Sugiyama, M.: Learning from complementary labels. In: NeurIPS. (2017)
10. Jiang, L. and Zhou, Z. and Leung, T. and Li, L. and Li, F.: MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In: ICML. (2018)
A Definition of Noise

The definition of the label transition matrix $T$ is as follow, where $\tau$ is the noise rate and $n$ is the number of the classes.
Pair flipping: \( T = \begin{bmatrix}
1 - \tau & \tau & 0 & \ldots & 0 \\
0 & 1 - \tau & \tau & \ldots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\tau & 0 & \ldots & 0 & 1 - \tau 
\end{bmatrix}, \)

Symmetry flipping: \( T = \begin{bmatrix}
1 - \tau & \frac{\tau}{n-1} & \ldots & \frac{\tau}{n-1} & \frac{\tau}{n-1} \\
\frac{\tau}{n-1} & 1 - \tau & \frac{\tau}{n-1} & \ldots & \frac{\tau}{n-1} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\frac{\tau}{n-1} & \ldots & \frac{\tau}{n-1} & 1 - \tau & \frac{\tau}{n-1} \\
\frac{\tau}{n-1} & \ldots & \frac{\tau}{n-1} & \frac{\tau}{n-1} & 1 - \tau 
\end{bmatrix}. \)

Note that pair flipping case is much harder than symmetry flipping case. For example, in Figure 4(a), the true class only has 10% more correct instances over wrong ones. However, the true has 37.5% more correct instances in Figure 4(b).

Fig. 4. Transition matrices of different noise types (using 5 classes as an example) [7].

B Network Structures

For MNIST, 28×28 gray image, the structure is 9-layer CNN. We also summarize it into Table 2.

For CIFAR-10, 32×32 RGB image, the structure is ResNet-32.

For NEWS, the structure is 3-layer MLP. We also summarize it into Table 3.

C Empirical results on NEWS

Results of PumpoutSL and MentorNet. Figure 5 shows test accuracy (top) and label precision (bottom) vs. number of epochs on NEWS dataset. On test accuracy, we can see PumpoutSL stops the memorization effects of deep networks to some degree. Especially on the harder Pair-45% and Symmetric-50% cases, PumpoutSL obviously achieves the higher accuracy over MentorNet along with
Table 2. 9-layer CNN used in our experiments on MNIST.

| CNN on MNIST |
|--------------|
| 28x28 Gray Image |
| 3x3 conv, 128 LReLU |
| 3x3 conv, 128 LReLU |
| 3x3 conv, 128 LReLU |
| 2x2 max-pool, stride 2 |
| dropout, $p = 0.25$ |
| 3x3 conv, 256 LReLU |
| 3x3 conv, 256 LReLU |
| 3x3 conv, 256 LReLU |
| 2x2 max-pool, stride 2 |
| dropout, $p = 0.25$ |
| 3x3 conv, 512 LReLU |
| 3x3 conv, 256 LReLU |
| 3x3 conv, 128 LReLU |
| avg-pool |
| dense 128→10 |

Table 3. 3-layer MLP used in our experiments on NEWS.

| MLP on NEWS |
|--------------|
| 300-D Embedding |
| dense 300→300, Softsign |
| dense 300→300, Softsign |
| dense 300→2 |

the training epochs. On label precision, while Standard fails to find clean instances again, Pumpout$_{SL}$ can achieve this especially on the hardest case due to the usage of scaled stochastic gradient ascent.
Fig. 5. Results of PumpoutSL and MentorNet on NEWS dataset. Top: test accuracy vs. number of epochs; bottom: label precision vs. number of epochs.

Results of PumpoutBC and nnBC Figure 6 shows test accuracy vs. number of epochs on NEWS dataset. Our PumpoutBC fully stops the decreasing process in two harder Pair-45% and Symmetry-50% cases, and effectively alleviates the decreasing process in one easier Symmetry-20% case. Meanwhile, in the hardest Pair-45% case, our PumpoutBC works better and better along with the training epochs. In this hardest case, our PumpoutBC finally achieves the higher accuracy over both BC and nnBC, although its accuracy falls behind BC in the first 90 epochs and nnBC in the first 50 epochs. Besides, in two symmetry cases, our PumpoutBC obviously achieves the higher accuracy over both BC and nnBC.

Fig. 6. Results of PumpoutBC and nnBC on NEWS dataset. Test accuracy vs. number of epochs.