Learning Fast Sample Re-weighting Without Reward Data

Zizhao Zhang Tomas Pfister
Google Cloud AI

Abstract

Training sample re-weighting is an effective approach for tackling data biases such as imbalanced and corrupted labels. Recent methods develop learning-based algorithms to learn sample re-weighting strategies jointly with model training based on the frameworks of reinforcement learning and meta learning. However, depending on additional unbiased reward data is limiting their general applicability. Furthermore, existing learning-based sample re-weighting methods require nested optimizations of models and weighting parameters, which requires expensive second-order computation. This paper addresses these two problems and presents a novel learning-based fast sample re-weighting (FSR) method that does not require additional reward data. The method is based on two key ideas: learning from history to build proxy reward data and feature sharing to reduce the optimization cost. Our experiments show the proposed method achieves competitive results compared to state of the arts on label noise robustness and long-tailed recognition, and does so while achieving significantly improved training efficiency. The source code is publicly available at https://github.com/google-research/google-research/tree/master/ieg.

1. Introduction

The performance of DNNs is dependent on the scale of training datasets and the quality of labels. Data biases are inevitable in practice, and in particular, noisy labels [51, 61] or imbalanced classes [9, 25] can negatively influence the model performance.

Sample re-weighting is an effective strategy that has been explored to address problems caused by data biases [5]. The underline principle of sample re-weighting is as simple as upgrading the weights of good samples and downgrading the weights of bad samples. Finding effective weights that can optimize the model training with stochastic gradient descent (SGD) is a dynamic process. The optimal weight of a sample can change over model training phases. Over-weighting simple samples at later phases while under-weighting hard samples at early phases can cause negative effects to the overall DNN accuracy [16]. In light of the advances in meta learning and reinforcement learning (RL), there is a growing interest in learning-to-learn based algorithms to optimize sample weights jointly with model training [44, 47, 20, 57]. This problem well resembles the design of MAML [13]: incorporating meta optimization inside the supervised training for sample weight optimization. The meta-objective of re-weighting is usually defined as finding the optimal weight per sample such that the trained model has the best objective on an additional reward (a.k.a. validation) dataset. Moreover, such reward dataset is required to be unbiased and to have reasonable size. For example, in label noise robust training problems, this unbiased reward dataset is expected to have clean and class-balanced labels. This extra requirement has been noted to be problematic, but how to remove such a requirement remains unanswered [6].

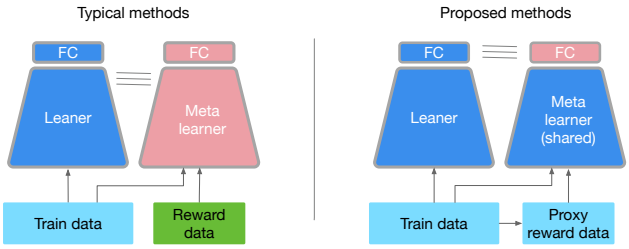

Figure 1. Overview of the proposed method compared with typical method for meta re-weighting. Our method is fast and does not need reward data.

From an optimization and efficiency perspective, although this problem can be directly formulated as an RL problem, the training computation is very expensive [15, 57]. Therefore, most existing work follows the more efficient meta learning framework, assuming that the weight optimization with respect to reward signals is a fully differentiable problem. Even so, similarly to MAML, the overall computation still requires a second-order unroll of DNN computation graphs, which increases the memory requirement and training time complexity significantly [44]. Such a limitation hinders the applicability of the method to large-scale DNNs, and emphasizes the importance of the need for further improving efficiency.

In this paper, we present a new fast sample re-weighting (FSR) method to overcome the two aforementioned problems

(Figure 1): a) removing reward data dependence and b) improving training efficiency. To this end we make the following contributions:

- We leverage a dictionary (essentially an extra buffer) to monitor the training history reflected by the model updates during meta optimization periodically, and propose a valuation function to discover meaningful samples from training data as the proxy of reward data. The unbiased dictionary keeps being updated and provides reward signals to optimize sample weights.
- Motivated by an investigation we conducted into the mechanism of sample weight meta-objective, instead of maintaining model states for both model and sample weight updates separately, we propose to enable feature sharing for saving the computation cost used for maintaining respective states. The proposed method demonstrates significant improvement in training efficiency, which is a desirable feature for large-scale DNN training compared to previous learning-based sample weighting methods.
- Because our proposed method does not rely on additional reward data, we can directly apply it to tackle common data biases, including noisy labels and long-tailed recognition, as well as more challenging complex of these two types of label corruption. Since fast re-weighting ability of FSR is orthogonal to domain-specific techniques, we also propose a momentum re-labeling technique with MixUp regularization to enhance the performance of FSR in noise robust training. Extensive experiments demonstrate our competitive performance compared to previous methods.

2. Related Work

Training samples are unequally important. Weighting training samples is a traditional and effective strategy to improve model performance. Traditional ML methods, such as importance sampling [24], AdaBoost [14], and self-paced learning [28], explore the usage of important samples for better model training. These weighting strategies have been extended to deep learning [22, 26, 35, 54, 5]. More specialized sample weighting approaches have also been developed in order to weight training samples targeting different goals, either to enhance high value data points, or to reduce data biases (e.g., label noises) [12, 27]. Hard sample mining [46] seeks for hard objects to bootstrap model training. Focal Loss [33] is widely used to allow DNNs to focus on hard samples. [9] proposes a class balanced (CB) loss for long-tailed recognition. [45] utilizes the softmax temperature to learn instance and class weights. Moreover, learning based re-weighting methods becomes popular for label noise-robust training. For example, [23] proposes to train a sequence model to predict good sample weighting. [44, 47, 20] propose meta learning based re-weighting. Besides learning sample weights alone, followup methods have explored learning additional data related coefficients, such as labels [61, 41, 31] and data augmentation policies [20].

Most of the popular learning based weighting methods originated from meta learning (or learning-to-learn) [19], e.g. MAML [13]. A common requirement is the access to a reward dataset that needs to be unbiased, so a meta-objective can be defined to optimize the sample weights. This reward dataset is mandatory for conventional meta learning used for few shot learning which requires models to quickly adapt to new tasks. However, in the task of learning sample weights, where reward data is drawn from the same distribution of training data, we question and investigate the necessity of that. Moreover, this type of methods consumes extensive compute and memory. Recently, the meta learning community has explored how to accelerate training of MAML [43, 39, 62]. However, to our best knowledge, no previous work has studied how to accelerate the training process for learning sample weights.

3. Background: Learning Sample Weights

In this section, we briefly review the background of learning based sample re-weighting methods [44, 47, 20, 10].

We define a dataset $D = \{(x_i^D, y_i^D), 1 \leq i \leq N\}$ with totally N samples and label y contains a certain degree of biases. Assuming we have another unbiased reward dataset $R = \{(x_i^R, y_i^R), 1 \leq i \leq M\}$ with totally M samples (where $M \ll N$). The objective of training DNN parameters Θ can be formulated as a weighted cross-entropy softmax loss,

$$\Theta^\star(\omega) = \arg\min_{\Theta} \sum_{i=1}^{N} \omega_i L\big(y_i^D, f(x_i^D; \Theta)\big), \qquad (1)$$

where ω_i is the sample loss weight for x_i^D. $f(\cdot;\Theta)$ is the targeting DNN that outputs class logits and $L(\cdot,\cdot)$ is the standard softmax cross-entropy loss for each training data pair (x, y). In regular supervised training, ω is equally distributed to samples in each mini-batch.

Here ω is treated as learnable parameters for optimization. To this end, the meta learning is formulated to learn the optimal ω for each training data in D, such that the trained model with new sample weights can perform best on the reward data in R, measured as the reward signal by a cross-entropy loss

$$\omega^\star = \arg\min_{\omega, \omega \geq 0} \frac{1}{M} \sum_{i}^{M} L\big(y_i^R, f(x_i^R; \Theta^\star(\omega))\big). \qquad (2)$$

The problem can be solved by enumerating real value sample weights in a brute-force fashion and training the model until converge at each weight combination. However, the computation requirement is infinite. Currently methods that try to

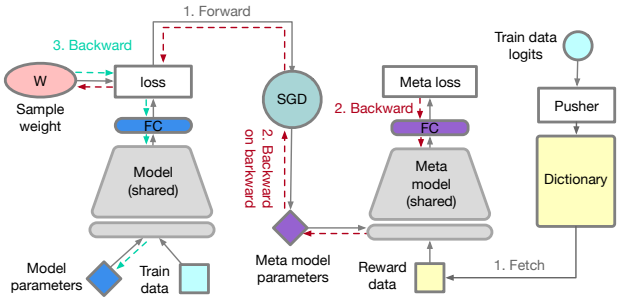

Figure 2. Illustration of the three major steps of the method. A dictionary is dynamically updated to maintain the proxy of reward data to enable meta optimization (gradient-by-gradient) of sample weight. Meta model parameters only include partial model parameters (e.g. FC in this example), while the rest are shared.

address this problem borrow the concept of meta learning to perform a single step model gradient update to online estimating $\Theta^\star$: $\Theta_{t+1}(\omega) = \Theta_t - \eta\nabla_\Theta\big(\sum_i \omega_i L(y_i, f(x_i;\Theta_t))\big)$, where η is a scalar step size. This enables differentiability of sample weight variables. Therefore, at each timestamp t, we can find the current optimal weight for each sample $\omega^\star_{t,i}$ through

$$\omega_{o,i} - \alpha\frac{1}{M}\sum_{i=1}^{M}\frac{\partial}{\partial\omega_{t,i}}L\big(y_i^R, f(x_i^R;\Theta_{t+1}(\omega))\big)\Big|_{\omega_{t,i}=\omega_o} \tag{3}$$

where α is the step size and w_o is the initial value. [44] resets $w_o = 0$ and calculates a new value every iteration. [47] treats it as a trainable variable and updates it using SGD. Lastly, the final weights $\omega^\star$ are normalized along mini-batch to satisfy $\sum_i \omega_i = 1$.

Training Complexity. $\Theta(\omega)$ is a function of ω, the gradient computation of ω in Equation (3) requires second-order back-propagation (also called gradient-by-gradient in literature). The whole training step requires 1) a forward pass on the training data and 2) reward data once each, 3) a step of gradient descent, 4) second-order back-propagation once, and 5) final model update with SGD. Therefore, there needs $3\times$ training time than regular supervised training as analyzed by [44]. At the same time, in modern deep learning libraries, the auto-differentiation mechanism will need to hold intermediate representations for second-order back-propagation, which takes higher GPU memory. The experimental section conducts detailed analysis.

4. Method

This section introduces the proposed method. Algorithm 1 presents the complete training details of FSR.

4.1. Learning from Past as Dictionary Fetching

Instead of preparing an extra reward set, we propose to use a dictionary to store valuable training samples that can be

Algorithm 1 FSR training step. The dictionary is updated every epoch. Θ^{meta} is pre-defined based on DNN architectures.

Input: Training data D, model parameter Θ, batch size b, dictionary R, reward batch size q, warm-up epoch E.
Initialize $R_0 = \text{BalancedSampleBatch}(D, |R|)$.
for $t = 1$ **to** $T-1$ **do**
  $\{X^D, Y^D\} \leftarrow \text{SampleBatch}(D, b)$.
  if $\text{Epoch}(t) \geq E$ **then**
    $\Theta^{\text{meta}}_t \leftarrow$ Synchronize with Θ_t.
    Initialize $\omega_o \leftarrow \frac{1}{b}$.
    $\Theta^{\text{meta}}_{t+1} = \Theta^{\text{meta}}_t - \alpha\nabla_{\Theta^{\text{meta}}}\sum_i^{|X^D|}\omega_i L\big(y_i, f(x_i;\Theta_t)\big)$.
    $\{X^R, Y^R\} \leftarrow \text{FetchBalancedBatch}(R_t, q)$.
    $\omega^\star \leftarrow$ Update sample weights (Equation 2).
    $\omega^\star \leftarrow \text{Normalize}(\omega^\star)$.
  else
    $\omega^\star \leftarrow \omega_o$
  end if
  $\Theta_{t+1} \leftarrow$ Update model given $\omega^\star$ (Equation 1).
  $\mathcal{P}(x, \Theta_{t+1}) \leftarrow$ Compute scores for $x \in X^D$ (Equation 5).
  $\hat{\mathcal{P}}_t(x) \leftarrow \lambda\hat{\mathcal{P}}_t(x) + (1-\lambda)\mathcal{P}(x, \Theta_{t+1}),\ x \in X^D$.
  if $\text{EpochEnd}(t)$ **then**
    $R_{t+1} \leftarrow$ Update dictionary using $\hat{\mathcal{P}}_t(D)$ (Equation 4).
  else
    $R_{t+1} \leftarrow R_t$
  end if
end for

used as a proxy of unbiased reward data, where data biases are controllable since labels and model predictions in R are known. The dictionary is dynamically updated to improve its quality. To this end, we maximize a defined a pusher function $\mathcal{P}$ such that the selected dictionary is

$$R^\star = \arg\max_R \sum_{x\in R, R\subset D}\mathcal{P}(x, \Theta), \tag{4}$$

where R has a fixed buffer size as an hyper-parameter. See experiments for more discussions about the memorization assumption behind the definition.

Choosing an effective $\mathcal{P}$ is important for the quality of reward data. This problem connects to an active research field on data valuation [15]. Different from these methods, the valuation calculation here needs to be efficient in order to execute with model updating at every step. We propose the following *meta-margin* definition

$$\mathcal{P}(x, \Theta_t) = L\big(f(x, \Theta_t), y\big) - L\big(f(x, \Theta^\star_t), y\big), \tag{5}$$

where $\Theta^\star_t$ represents model state after meta update at timestamp t. The proposed meta-margin utilizes the states between the model and the meta model on training data. Maximizing meta-margin intuitively finds samples whose losses have the largest drop after model gradient descent. It prioritizes training samples which is not well recognized (i.e., $L(f(x, \Theta_t), y)$ is high) but its loss can be well minimized

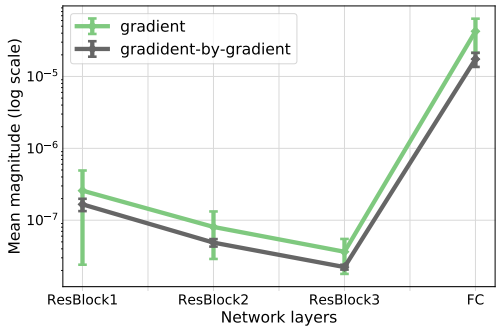

Figure 3. Gradient and gradient-by-gradient illustration of Wide-ResNet28-10 residual layer blocks.

after then. In contrast, if a sample has small loss (i.e., $L(f(x, \Theta_t), y)$ is low) already (possibly being memorized), the meta-margin will be small as well. More importantly, if the loss cannot be minimized or even increases after update, the pusher function de-prioritizes these samples because they are likely undesirable.

In label noise robust training, why the meta-margin can avoid selecting mislabeled (undesirable) samples? In practice, mislabeled data is usually harder to be minimized than clean data due to the regularization of DNNs that is able to resist label noise. This phenomenon is prominent at early training phases [51, 34] because models tend to learn simple samples before hard (mislabeled) samples [45, 3]. It is worth noticing that utilizing this intrinsic regularization to deal with noisy labels is fairly common in previous methods to divide possible clean and noisy samples [42, 17, 34, 11]. For example, [42] proposes to calculate margin between logits of the labeled class and largest logits of other classes to achieve this aim. Our proposed meta-margin borrows this high-level inspiration, while at the same time makes use of the meta optimization behaviors to avoid several types of samples that have low value as proxy reward data. In experiments, we also explore several existing data valuation candidates as the pusher function to verify the effectiveness of the proposed meta-margin.

Momentum pusher. The pusher function $\mathcal{P}$ depends on the state of models. To obtain a robust estimation of data points using a series of history of model states, we propose a momentum based pusher function, where $\hat{P}_t = \lambda \hat{P}_{t-1} + (1-\lambda)P_t$. Therefore, we construct R as the reward dataset using the momentum version $\mathcal{P}$ of Equation (4). Lastly, at the start of the training when model is poorly performed, the constructed dictionary could be noisy. Thus we usually setup a couple of epochs for warm-up without applying learned weights (see Algorithm 1 for details).

4.2. Feature Sharing to Reduce Training Time

Efficiency is an well-known bottleneck for learning based re-weighting, because it requires meta-learning-like nested updates. The experimental section shows quantitive analysis.

We show that improving the training efficiency needs a revisit of model parameter behaviors between nested outer and inner loops. We speculate the potential possibility of feature sharing between Θ_{t+1} and Θ_t. To be specific, Equation 3 can be further written as

$$\begin{aligned}
&\frac{\partial}{\partial \omega_{t,i}} L^R \Big|_{\omega_{t,i}=\omega_o} \\
&= \frac{1}{M}\sum_{j=1}^{M} \frac{\partial L_j^R}{\partial \Theta}\Big|_{\Theta=\Theta_{t+1}} \frac{\partial \Theta_{t+1}(\omega)}{\partial \omega_{t,i}}\Big|_{\omega_{t,i}=\omega_o} \\
&\propto \frac{1}{M}\sum_{j=1}^{M}\sum_{l\in\Theta} \frac{\partial L_j^R}{\partial \Theta_l}\Big|_{\Theta_l=\Theta_{t,l}} \frac{\partial L_i^D}{\partial \Theta_l}\Big|_{\Theta=\Theta_{t,l}} \\
&\propto \frac{1}{M}\sum_{j=1}^{M}\sum_{l\in\Theta^{\text{meta}}} \frac{\partial L_j^R}{\partial \Theta_l}\Big|_{\Theta_l=\Theta_{t,l}} \frac{\partial L_i^D}{\partial \Theta_l}\Big|_{\Theta=\Theta_{t,l}},
\end{aligned} \tag{6}$$

where Θ_l refers to the l-th layer of parameters. The derivation indicates the meta-gradient is the sum of the gradient products of all layers of parameters. It motivates us to use partial layers $\Theta^{\text{meta}} \subset \Theta$ to approximate Equation (6) (i.e. the last row). To understand the contribution of different layers, Figure 3 shows the magnitudes of gradient-by-gradient $\nabla_{\Theta_{t+1}} L^R$ and gradient $\nabla_{\Theta_t} L^R$ of several layers, suggesting that the changes of sample weights are majorly contributed by the fully-connected (FC) layer. Thus including FC into Θ^{meta} could be sufficient to achieve good approximation with the lowest computation cost.

This approximation is actually equivalent to enabling feature sharing between the model and the meta model, which connects to the recent investigation of rapid learning in MAML [43]. In implementation, it is achieved by excluding layers of parameters from bottom up to a certain layer from participating meta optimization (see Figure 2). For example, if only FC is included, although a reward data mini-batch still needs to forward the meta model, all rest meta gradient-by-gradient computation is reduced to cheap matrix (from FC weights) multiplication.

4.3. Momentum Re-labeling and Regularization

Beside re-weighting, we explore more task specific component to improve FSR on noise robust training. Recent methods favor the design that identifying mislabeled samples and then reusing them as unlabeled data for re-labeling and augmentation [11, 30], in light of semi-supervised learning [4, 48, 55, 49, 63]. FSR intrinsically assigns low weight values to filter mislabeled samples from participating into supervision. To enhance FSR for better reuse of those samples, we propose a momentum dictionary to maintain long-term prediction estimation of samples and use the estimated predictions as pseudo labels for supervised training. Compared with previous methods with re-labeling, our approach is

Table 1. Test accuracy on CIFAR10 with uniform noise. ♯ indicates methods requires additional reward data. ±0.0 and '-' mean the corresponding method does not report the value.

Method	Noise ratio			
	0	0.2	0.4	0.8
GCE	93.5	89.9±0.2	87.1±0.2	67.9±0.6
RoG	94.2	87.4±0.0	81.8±0.0	-
MentorNet♯	96.0	92.0±0.0	89.0±0.0	49.0±0.0
L2R♯	96.1	90.0±0.4	86.9±0.2	73.0±0.8
MWN♯	92.0	90.3±0.6	87.5±0.2	-
CRUST	94.4	91.1±0.2	89.2±0.2	58.3±1.8
ELR	94.5	92.1±0.4	91.4±0.2	80.7±0.6
FSR-R32	94.4	91.8±0.7	90.2±0.7	74.2±0.9
FSR	96.8	**95.1±0.1**	**93.7±0.1**	**82.8±0.3**

Table 2. Test accuracy on CIFAR100 with uniform noise.

Method	Noise ratio			
	0	0.2	0.4	0.8
GCE	81.4	66.8±0.4	61.8±0.2	47.7±0.7
MentorNet♯	79.0	73.0±0.0	68.0±0.0	35.0±0.0
L2R♯	81.2	67.1±0.1	61.3+2.0	35.1±1.2
MWN♯	70.1	64.2±0.3	58.6±0.5	-
PENCIL	81.4	73.9±0.3	69.1±0.6	-
ELR	75.2	74.7±0.3	68.4±0.4	30.2±0.8
FSR-R32	71.3	69.8±0.2	65.9±0.2	41.2±3.0
FSR	81.6	**78.7±0.2**	**74.2±0.4**	**46.7±0.8**

fairly simpler since it does not need extra copies of extensively augmented data to ensemble pseudo labels or two-stage training to bootstrap mislabeled data [32, 30, 11].

The pseudo label $\hat{y}$ of a sample x at timestamp t is updated in a moving-average manner

$$\hat{y}_t = \beta \hat{y}_{t-1} + (1-\beta) f(x, \Theta_t), \tag{7}$$

where β is moving-average decay scalar. The batch samples with estimated labels are simply used to construct an extra softmax loss with a multiplier p with the original weighted loss, so the total loss is $\sum_{x_i} \omega_i L(y_i, f(x_i)) + p \cdot L(\hat{y}_i, f(x_i))$.

Regularization. In addition, we apply MixUp [59] on training data used to compute the weighted loss only (i.e., the first term of the total loss). This technique almost becomes a common practice for noise robust methods [50]. In experiments, we find MixUp has strong regularization effects to improve dictionary quality (i.e. label clean ratio) for noise robust training.

5. Experiments

5.1. Label Noise Experiments

We test our method on the CIFAR and WebVision datasets. The compared methods including GCE [60], MentorNet [23], RoG [29], L2R [44], MWN [47], F-correction [40], D2L [37], Co-teaching [17], Iterative-CV [7], PENCIL [56], F-Correction [1], CRUST [38], and ELR [34].

Table 3. Test accuracy on CIFAR10 with asymmetric noise.

Method	Noise ratio	
	0.2	0.4
GCE	89.5±0.3	82.3±0.7
F-Correction	89.1±0.5	83.6±0.3
PENCIL	92.4±0.0	91.2±0.0
L2R-R32	89.2 ±0.3	84.8±0.0
L2R	92.4 ±0.1	90.8±0.3
FSR-R32	91.5±0.1	90.2 ±0.1
FSR	**95.0±0.1**	**93.6±0.3**

Table 4. Top: Hyper-parameters used by different experiments, where most are fixed across experiments. Some hyper-parameters are defined in Algorithm 1. Bottom: Accuracy with sweeping hyper-parameters on CIFAR100 with 40% uniform noise.

Experiment	Hyper-parameter							
	$\|R\|$	q	b	λ	η	α	β	p
CIFAR (noise)	2k	200	100	0.9	0.1	1	0.1	2
Webvision	5k	200	16	0.9	0.1	1	0.1	1
CIFAR (Long-Tailed)	3k	800	100	0.9	0.1	1	-	-
iNaturalist2018	2k	350	32	0.9	0.1	1	-	-
CIFAR (LT+noise)	3k	200	128	0.9	0.1	1	0.1	2

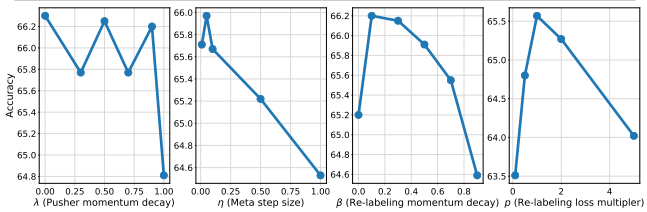

Table 5. Test accuracy on WebVision (50 classes). ImageNet validation accuracy on the 50 classes is also reported.

Method	ImageNet		WebVision	
	top1	top5	top1	top5
F-correction	57.4	82.4	61.1	82.7
D2L	57.8	81.4	62.7	84.0
Co-teaching	61.5	84.7	63.6	85.2
Iterative-CV	61.6	85.0	65.2	85.3
MentorNet♯	63.8	85.8	63.0	81.4
CRUST	67.4	**87.8**	72.4	**89.6**
FSR	**72.3**	87.2	**74.9**	88.2

CIFAR. We first verify on the common CIFAR10 and CIFAR100 uniform label noises following settings of [44, 23]. We use WRN28-10 as default [58] as it is commonly used in [44, 47]. We also test using ResNet32 [18]. We train the model on a single V100 GPU. We use a cyclical cosine learning rate for 128 epochs and report the accuracy corresponding to the last iteration.

Table 6. Test accuracy on CIFAR long-tailed recognition. FSR results are obtained by averaging 3 runs. DF refers to deferred enabling re-weighting. CB refers to class-balanced loss with different sub-types [9]. CB-Best adopts the best hyper-parameters for each setup. Note that ♯ indicates methods uses 10 image per class as a reward set, which is not absolute fair comparison.

Dataset	CIFAR10			CIFAR100		
Imb. ratio	200	50	10	200	50	10
SoftMax	65.68	74.81	86.39	34.84	43.85	55.71
CB-Focal	65.29	76.71	86.66	32.62	44.32	55.78
CB-Best	68.89	79.27	87.49	36.23	45.32	57.99
L2R$^\sharp$	66.51	78.93	85.19	33.38	44.44	53.73
MWN$^\sharp$	**68.91**	**80.06**	87.84	**37.91**	**46.74**	**58.46**
FSR-DF	66.15	79.78	**88.15**	36.74	44.43	55.60
FSR	67.76	79.17	87.40	35.44	42.57	55.45

Table 1 and Table 2 show the results on the two datasets with different noise ratios, respectively. The proposed FSR outperforms compared methods, and also learning based weight competitors, i.e. L2R, MentorNet, and MWN, which requires additional clean reward data. We also verify on asymmetric noise types, which confuses visually similar categories. Table 4 specifies and studies the hyper-parameters used by different experiments. As we can seen, the biggest change by sweeping these hyper-parameters is $\sim 2\%$, indicting insensitivity to hyper-parameters. Dictionary size and reward data batch size needs to be changed for different datasets since a class-balanced dictionary size and reward data mini-batch depends to the number of classes.

WebVision. Then, we scale up FSR to large-scale WebVision dataset with 50 classes [23], which contains the top-50 categories of ImageNet. Following the compared methods, ResNet50 is used with random initialization. We use the same training schedule used by CIFAR experiments above. Differently, we train on 8 GPUs for 128 epochs. Table 5 compares results to previous methods, demonstrating promising results including the best top-1 accuracy, although the compared CRUST has higher top-5 accuracy. FSR shows particular strong generalization ability to ImageNet validation accuracy.

5.2. Long-tailed Imbalance Experiments

CIFAR. We test FSR on long-tailed CIFAR benchmarks, with different imbalance ratios as defined by [9]. We modify some training configurations as we use for label noise experiments. First, we only apply original FSR to test the ability of re-weighting (i.e. without momentum re-labeling and MixUp). Second, similar like related methods [9], we apply 0.1 softmax label smoothing to compute the loss for re-weighting. Third, Equation (2) constraints weight to be non-negative and clips negative values $\max(0, \omega^\star)$ before normalization. Here we replace this constraint by shifting all weights to be non-zero $(\omega^\star - \min(\omega^\star) + \frac{1}{b})$ before normalization. Fourth, recent methods [6, 25] commonly apply deferred balancing, which trains model in the supervised manner to learn representations and then apply re-balancing methods to fine-tune the model. We reuse the 200-epoch learning schedule from [9] (see Table 4 for extra hyper-parameter details). We optionally apply this deferred schedule by enabling FSR after the first learning rate decay at 160 epoch (i.e., set the warm-up epoch $E = 160$). We report results with and without the deferred schedule in Table 6. Overall, FSR achieves competitive results to other methods, although it marginally underperforms MWN, which requires additional balanced reward data.

Table 7. Results with mixed label corruption in CIFAR10. Uniform noise ratios are added onto different imbalance ratios.

Noise ratio	0.2				0.4			
Imbal. ratio	0	10	50	200	0	10	50	200
CRUST [38]	90.2	65.7	41.5	34.3	89.2	59.5	32.4	28.8
FSR (Ours)	91.8	85.7	77.4	65.5	90.2	81.6	69.8	49.5

iNaturalist. We also test our method on iNaturalist 2018 [21], a large-scale long-tailed recognition benchmark. Following the 90 epoch learning schedule of [9] with deferred schedule, FSR achieves promising results compared to previous methods: 65.52% top-1 (85.02% top-5) accuracy, against CB Focal [9] $61.12\%(81.0\%)$ and Remix [8] 61.31% (82.30%). Considering FSR here is merely a generic re-weighting approach, we believe it can be incorporated with more sophisticated designs for long-tailed recognition to obtain further improvement.

5.3. Complex of Label Noise and Imbalance

Although label noise and imbalance are usually studied as independent research, in real-world applications, these label corruption actually happen simultaneously. This is a more realistic yet challenging setup. We take an initiative to tackle this problem. To benchmark, we add uniform noise onto imbalanced CIFAR10. Table 7 compares the results with the best-performing noise-robust method CRUST [38]. It can be seen that FSR achieves much higher improvement margin over compared methods on this dataset than that on the noise-robust datasets, suggesting the robustness of FSR to complex label corruption.

5.4. Training Complexity

Recent learning based sample weighting methods have similar theoretical training complexity [44, 20, 47]. We use L2R for direct comparison[1]. Figure 4(left) compares the training time and GPU memory cost on three architectures[2].

[1]We re-implement the method using the same programming technique for gradient-by-gradient.

[2]TensorFlow profiler toolkit https://www.tensorflow.org/guide/profiler

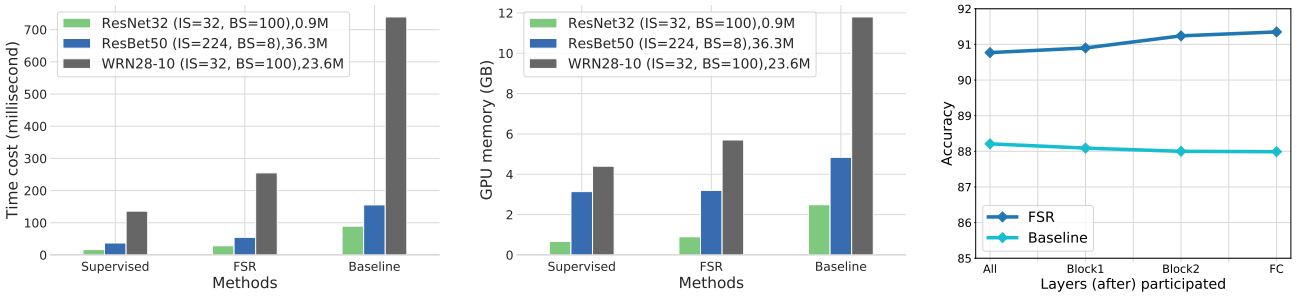

Figure 4. Left: Training complexity comparison in terms of time per second (left) and GPU memory (right). Three network architectures (with different numbers of parameters) are profiled with certain input image size (IS) and batch size (BS). Right: Accuracy with different numbers of layers included in Θ^{meta} for learning sample weights (ResNet32 is used here). x-axis indicates the layers after the denoted layer are participated into meta optimization. For instance, FC indicates only the softmax layer is participated (i.e. our default setting)

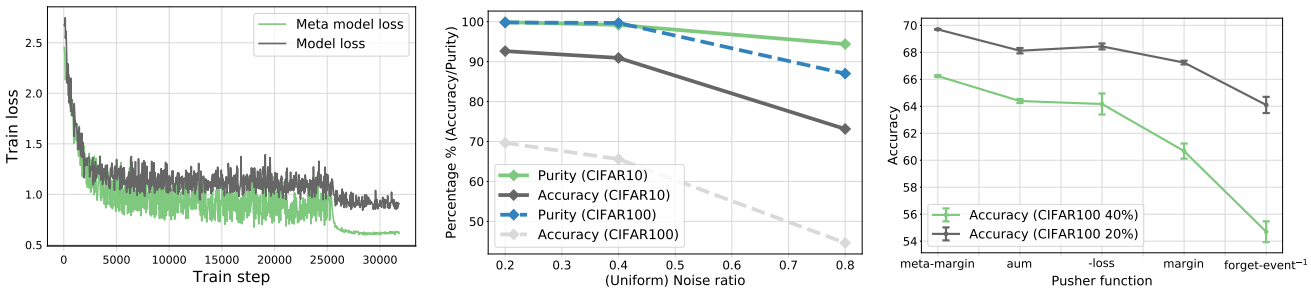

Figure 5. Ablation study. See explanation in main text. Left: Train loss visualization of model Θ_t and meta model Θ_{t+1}. Middle: Dictionary purity versus accuracy under different datasets and noises ratios. Right: Comparison of different pusher functions on CIFAR100 with 20% and 40% uniform noise.

Compared to regular supervised training, FSR increases the training cost by $47\% \sim 87\%$ while the baseline increases $319\% \sim 443\%$. For GPU memory usage, FSR increases the memory overhead by $1.9\% \sim 34\%$ while the baseline increases $54\% \sim 270\%$. The results suggest that FSR significantly improves the training efficiency. In addition, FSR requires one-stage training only so it is expected to consume less training cost than popular multi-stage methods, such as our compared Co-teaching [17] and Iterative-CV [7].

Furthermore, we study how Θ^{meta} impacts the final performance. Does strong feature sharing sacrifice accuracy? We control the number of layers included in Θ^{meta}. Figure 4(right) reports accuracy on CIFAR10 40% with uniform noise using ResNet32, suggesting that using partial layers does not impact much on either L2R or FSR. It is interesting to note that, for FSR, including fewer layers consistently leads to higher accuracy. The study indicates a strong feature sharing is feasible between outer loop and inner loop of a meta optimization step.

5.5. More Studies and Discussions on FSR

Meta model memorization. Memorization is a factor we need to consider to avoid trivial solution. If all training data are memorized and generates zero training loss, $\mathcal{P}(x, \Theta)$ in Equation (4) will contain no useful information. However, it is unlikely for a well-regularized DNNs to memorize (overfit) the training dataset. Figure 5(left) shows the train loss of model Θ_t and meta model $\Theta_t^\star$ at every timestamp on long-tailed CIFAR (imbalance ratio $= 10$). Since meta model is a 'locally' optimized model, it leads to lower softmax loss in average, yet no phenomenon of over-fitting. This observation also applies to all other datasets and architectures we experimented.

Dictionary purity. In the label noise task, the purity (the clean ratio) of dictionary R plays a critical role in model performance. If undesirable samples are pushed into the dictionary, accuracy can be affected. Figure 5(middle) visualizes the accuracy versus dictionary purity under different noise ratios. As can been seen, dictionary purity at 80% noise ratio (CIFAR100 and CIFAR100 both) reduces clearly and thereby causes clear accuracy drop.

We further study a variety of pusher function alternatives against the proposed meta-margin. The simplest *negative-loss* prioritizes well-recognized samples. *max-margin* is a popular method in active learning [2] which we use here to select certain samples. *forgetting-event* [53] finds easy-to-forget samples as they are bad or hard samples. A high forgetting rate indicates corrupted labels. *area-under-margin* (AUM) [42] produces wider margin on clean samples. As can been seen in Figure 5(right), the proposed meta-margin

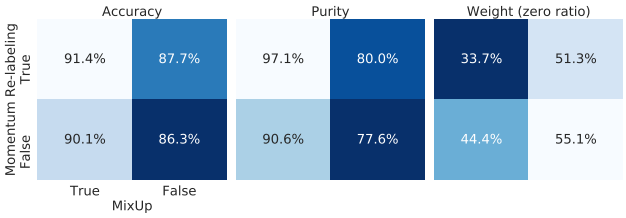

Figure 6. Accuracy, dictionary purity, and sample weight zero ratio studies on combinations of momentum re-labeling and MixUp, experimented on CIFAR10 with 20% uniform noise.

performs clearly better than negative loss and AUM, though we observe they can all select clean labels with high rates. Forget-event^{-1} performs the worst. We realize it is negatively impacted by MixUp. Removing MixUp will recover CIFAR100 20%/40% noise to accuracy 68.1%/60.9%.

Impact of momentum re-labeling and MixUp. We study how MixUp and the proposed momentum re-labeling (Section 4.3) affect FSR. In Figure 6, we show the accuracy, dictionary purity, and also averaged ratio of zero (inactive) weights. The ratio of zero weight is expected to be close to the noise ratio of the dataset. Higher ratios of zero weights than actually noise ratios hinder sufficient supervision while lower ratios introduces noisy supervision. MixUp has great impact to the dictionary purity, and therefore final accuracy. As we can seen in Figure 6 (right), the ratio of zero weights is closest to the noise ratio if MixUp is enabled. We think MixUp plays a particular regularization role than just augmentation. We do not find it is effective for long-tailed recognition. Label smoothing [36] is not an effective alternative either. We hypothesize it is related to the calibration effects of MixUp [52] which improves the pusher function and dictionary quality. Future work is useful for investigation. Furthermore, momentum re-labeling can further improve all metrics in Figure 6, and pave the last mile to lead accuracy in Table 1.

Reward data versus proxy dictionary. [3] The size of reward data in L2R [44] has impacts to the model performance. We train L2R with different reward data size on CIFAR10 with 40% uniform noise. As shown in Figure 7, L2R does not benefit much with more reward data and an over-large reward set can even hurt the accuracy. The accuracy of L2R also drops largely given very limited reward data. This observation is aligned with the findings by [44].

We then study the impact dictionary size on the accuracy of the proposed FSR. We conduct the same experiments for FSR and FSR without momentum re-labeling and MixUp (denoted as FSR-Raw). We find FSR is insensitive to the dictionary size and FSR-Raw demonstrates normal sensitivity

[3]There are totally 50k training data. In this controllable experiments, a fixed 45k training data is used for all methods. For L2R, the varying reward data is split from the rest 5k data for the different settings. FSR does not use extra data.

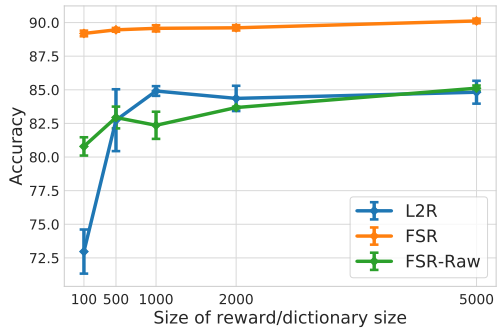

Figure 7. Results of the compared L2R, FSR, and FSR-Raw (without momentum re-labeling and MixUp) under varying reward data size (for L2R) and dictionary size (for FSR and FSR-Raw).

yet much better than the sensitivity of L2R to real reward data. Given sufficient dictionary size, FSR-Raw can perform as good as L2R with arbitrary reward data size. This study further suggests the actual unbiased reward data is dispensable and could be replaced by a well-picked training data subset when with sufficient DNN regularization.

6. Conclusion

This paper presents a fast sample re-weighting method, named FSR, that addresses two key bottlenecks for learning based sample weighting methods: high training cost and dependence on additional reward data. The fast re-weighting ability of FSR is orthogonal to domain-specific techniques. We have shown that by incorporating task specific components (the proposed momentum re-labeling and MixUp), FSR outperforms previous noise robust methods. We conduct extensive experiments to demonstrate its effectiveness on noisy labels and long-tailed class recognition benchmarks.

As future work, we think FSR has the potential to improve other tasks where sample re-weighting matters. In addition, we observe from experiments that MixUp has a significant impact on the performance of noisy robust training. Exploring more universal regularization techniques could potentially let FSR generalize better to other domains.

Acknowledgments

We would like to thank Chen-Yu Lee, Kihyuk Sohn, and Han Zhang for their valuable discussions.

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. *ICML*, 2019. 5

[2] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, 2007. 7

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009. 4

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, 2019. 4

[5] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *ICML*, 2019. 1, 2

[6] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019. 1, 6

[7] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. *ICML*, 2019. 5, 7

[8] Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. Remix: Rebalanced mixup. In *ECCV*, 2020. 6

[9] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019. 1, 2, 6

[10] Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. Learning to learn from weak supervision by full supervision. *NIPS Workshop on Meta-Learning*, 2017. 2

[11] Yifan Ding, Liqiang Wang, Deliang Fan, and Boqing Gong. A semi-supervised two-stage approach to learning from noisy labels. In *WACV*, 2018. 4, 5

[12] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *ICCV*, 2017. 2

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 1, 2

[14] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 2

[15] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, 2019. 1, 3

[16] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *ICML*, 2019. 1

[17] 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 *NeurIPS*, 2018. 4, 5, 7

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[19] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020. 2

[20] Zhiting Hu, Bowen Tan, Russ R Salakhutdinov, Tom M Mitchell, and Eric P Xing. Learning data manipulation for augmentation and weighting. In *NeurIPS*, 2019. 1, 2, 6

[21] iNaturalist. The inaturalist 2018 competition dataset. 2018. 6

[22] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. Self-paced curriculum learning. In *AAAI*, 2015. 2

[23] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *ICML*, 2018. 2, 5, 6

[24] Herman Kahn and Andy W Marshall. Methods of reducing sample size in monte carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953. 2

[25] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *ICLR*, 2020. 1, 6

[26] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *ICML*, 2018. 2

[27] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *TNNLS*, 2017. 2

[28] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 2

[29] Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. *ICML*, 2019. 5

[30] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *ICLR*, 2020. 4, 5

[31] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, 2019. 2

[32] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *ICCV*, 2017. 5

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2

[34] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *arXiv preprint arXiv:2007.00151*, 2020. 4, 5

[35] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *TPAMI*, 2015. 2

[36] Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *ICML*, 2020. 8

[37] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. *ICML*, 2018. 5

[38] Baharan Mirzasoleiman, Kaidi Cao, and Jure Leskovec. Coresets for robust training of neural networks against noisy labels. *NeurIPS*, 2020. 5, 6

[39] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2

[40] 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 *CVPR*, 2017. 5

[41] Hieu Pham, Qizhe Xie, Zihang Dai, and Quoc V Le. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020. 2

[42] Geoff Pleiss, Tianyi Zhang, Ethan R Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *NeurIPS*, 2020. 4, 7

[43] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *ICLR*, 2020. 2, 4

[44] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *ICML*, 2018. 1, 2, 3, 5, 6, 8

[45] Shreyas Saxena, Oncel Tuzel, and Dennis DeCoste. Data parameters: A new family of parameters for learning a differentiable curriculum. *NeurIPS*, 2019. 2, 4

[46] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 2

[47] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019. 1, 2, 3, 5, 6

[48] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020. 4

[49] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. *arXiv preprint arXiv:2005.04757*, 2020. 4

[50] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020. 5

[51] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018. 1, 4

[52] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *NeurIPS*, 2019. 8

[53] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018. 7

[54] Yixin Wang, Alp Kucukelbir, and David M Blei. Robust probabilistic modeling with bayesian data reweighting. In *ICML*, 2017. 2

[55] Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation. *NeurIPS*, 2020. 4

[56] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. *CVPR*, 2019. 5

[57] Jinsung Yoon, Sercan Arik, and Tomas Pfister. Data valuation using reinforcement learning. In *ICML*, 2020. 1

[58] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016. 5

[59] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR*, 2017. 5

[60] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018. 5

[61] Zizhao Zhang, Han Zhang, Sercan O Arik, Honglak Lee, and Tomas Pfister. Distilling effective supervision from severe label noise. In *CVPR*, 2020. 1, 2

[62] Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Caml: Fast context adaptation via meta-learning. *ICML*, 2019. 2

[63] Yuliang Zou, Zizhao Zhang, Han Zhang, Chun-Liang Li, Xiao Bian, Jia-Bin Huang, and Tomas Pfister. Pseudoseg: Designing pseudo labels for semantic segmentation. *ICLR*, 2021. 4