Learning Deep Neural Networks under Agnostic Corrupted Supervision

Boyang Liu [1] **Mengying Sun** [1] **Ding Wang** [1] **Pang-Ning Tan** [1] **Jiayu Zhou** [1]

Abstract

Training deep neural network models in the presence of corrupted supervision is challenging as the corrupted data points may significantly impact generalization performance. To alleviate this problem, we present an efficient robust algorithm that achieves strong guarantees without any assumption on the type of corruption and provides a unified framework for both classification and regression problems. Unlike many existing approaches that quantify the quality of the data points (e.g., based on their individual loss values), and filter them accordingly, the proposed algorithm focuses on controlling the collective impact of data points on the average gradient. Even when a corrupted data point failed to be excluded by our algorithm, the data point will have a very limited impact on the overall loss, as compared with state-of-the-art filtering methods based on loss values. Extensive experiments on multiple benchmark datasets have demonstrated the robustness of our algorithm under different types of corruption. Our code is available at https://github.com/illidanlab/PRL.

1. Introduction

Corrupted supervision is a common issue in real-world learning tasks, where the target variables are potentially noisy due to errors in the data collection or labeling process. Such corruptions can have severe consequences especially in deep learning models, whose large degree-of-freedom makes them easier to memorize the corrupted examples, and thus, susceptible to overfitting (Zhang et al., 2016).

There have been extensive efforts to achieve robustness against corrupted supervision. A natural approach to deal with corrupted supervision in deep neural networks (DNNs) is to reduce the model exposure to corrupted data points during training. By detecting and filtering (or re-weighting) the possible corrupted samples, the learning algorithm is expected to deliver a model that is similar to the one trained on clean (uncorrupted) data (Han et al., 2018; Zheng et al., 2020). There are various criteria designed to identify the corrupted samples in training. For example, Han et al. (2018); Jiang et al. (2018) leveraged the loss function values of the data points; Zheng et al. (2020) considered prediction uncertainty for filtering data; Malach & Shalev-Shwartz (2017) used the disagreement between two deep networks; while Reed et al. (2014) utilized the prediction consistency of neighboring iterations. The success of these methods highly depends on the effectiveness of the detection criteria in correctly identifying the corrupted data points. Since the actual corrupted points remain unknown throughout the learning process, such "unsupervised" methods may not be effective, either lacking in theoretical guarantees of robustness (Han et al., 2018; Reed et al., 2014; Li et al., 2017) or providing guarantees only under the assumption that prior knowledge is available about the type of corruption present (Zheng et al., 2020; Shah et al., 2020; Malach & Shalev-Shwartz, 2017; Patrini et al., 2017; Yi & Wu, 2019). Most existing theoretical guarantees under agnostic corruption during optimization are focused on convex losses (Prasad et al., 2018) or linear models (Bhatia et al., 2015; 2017), and thus cannot be directly applied to DNNs. Diakonikolas et al. (2019) developed a generalized non-convex optimization algorithm against agnostic corruptions. However, it is not optimized for the label/supervision corruption problem and has a high space complexity, which is prohibitively costly when applied to typical DNNs with a large amount of parameters. Furthermore, many existing approaches are exclusively designed for classification problems (e.g., Malach & Shalev-Shwartz (2017); Reed et al. (2014); Menon et al. (2019); Zheng et al. (2020)); extending them to solving regression problems is not straightforward.

To tackle these challenges, this paper presents a unified optimization framework with robustness guarantees, without any assumptions on how supervisions are corrupted, and is applicable to both classification and regression problems. Instead of designing a criterion for accurate detection of corrupted samples, we focus on limiting the collective impact of corrupted samples during the learning process through *robust mean estimation* of the gradients. Specifically, if our

[1]Department of Computer Science and Engineering, Michigan State University, USA. Correspondence to: Boyang Liu <liuboya2@msu.edu>, Jiayu Zhou <jiayuz@msu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

estimated average gradient is close to the expected gradient from the clean data during the learning iterations, then the final model will be close to the model trained on clean data. As such, a corrupted data point can still be used during the training as long as it does not significantly alter the average gradient. This observation has remarkably impacted our algorithm design: instead of explicitly quantifying (and identifying) individual corrupted data points, which is a hard problem in itself, we are now dealing with an easier task, i.e., eliminating training data points that significantly distort the mean gradient estimation. One immediate consequence of this design is that, even when a corrupted data point failed to be excluded by the proposed algorithm, the data point will likely have very limited impact on the overall gradient, unlike existing approaches that filter data points based on their loss values. Compared to state-of-the-art robust optimization methods (Prasad et al., 2018; Diakonikolas et al., 2019) that require the more expensive SVD computation on the gradient matrix, we fully utilize the gradient structure when the corruptions are restricted to the target variable to make our algorithm applicable to DNNs. Moreover, when only the target variable is corrupted, we improve the error bound from $\mathcal{O}(\sqrt{\epsilon})$ to $\mathcal{O}(\epsilon)$, where ϵ is the corruption rate. We perform experiments on both regression and classification tasks with corrupted supervision on multiple benchmark datasets. The results show that the proposed method outperforms various state-of-the-art baseline methods.

2. Background

Learning from corrupted data (Huber, 1992) has attracted considerable attention in the machine learning community (Natarajan et al., 2013). Many recent studies have investigated robustness of classification tasks with noisy labels. For example, Kumar et al. (2010) proposed a self-paced learning (SPL) approach, which assigns higher weights to examples with smaller loss. A similar idea was used in curriculum learning (Bengio et al., 2009), in which the model learns the easy concept first before the harder ones. Alternative methods inspired by SPL include weight learning (Jiang et al., 2018) and collaborative learning (Han et al., 2018; Yu et al., 2019) approaches. Label correction (Patrini et al., 2017; Li et al., 2017; Yi & Wu, 2019) is another approach, which attempts to revise the original labels of the data to recover clean labels from corrupted ones. However, since we do not have access to which data points are corrupted, it is harder to obtain provable guarantees for label correction without strong assumptions about the corruption type.

Accurate estimation of the gradient is a key step for successful optimization. The relationship between gradient estimation and its final convergence has been widely studied in the optimization community. Since computing an approximated (and potentially biased) gradient is often more efficient than computing the exact gradient, many studies used approximated gradients to optimize their models and showed that they suffer from the biased estimation problem if there is no assumption on the gradient estimation (d'Aspremont, 2008; Schmidt et al., 2011; Bernstein et al., 2018; Hu et al., 2020; Ajalloeian & Stich, 2020).

A closely related topic is robust estimation of the mean. Given corrupted data, robust mean estimation aims at generating an estimated mean $\hat{\mu}$ such that the difference between the estimated mean on corrupted data and the mean of clean data $\|\hat{\mu} - \mu\|_2$ is minimized. The median or trimmed mean have been shown to be optimal statistics for mean estimation in one-dimensional data (Huber, 1992). However, robustness in high dimension is more challenging since applying the coordinate-wise optimal robust estimator would lead to an error factor $\mathcal{O}(\sqrt{d})$ that scales with dimensionality of the data. Although classical methods such as Tukey median (Tukey, 1975) have successfully designed algorithms to eliminate the $\mathcal{O}(\sqrt{d})$ error, the algorithms cannot run in polynomial-time. More recently, Diakonikolas et al. (2016); Lai et al. (2016) successfully designed polynomial-time algorithms with dimension-free error bounds. The results have been widely applied to improve algorithmic efficiency in various scenarios (Dong et al., 2019; Cheng et al., 2020).

Robust optimization is designed to improve algorithm robustness in the presence of corrupted data. Most existing efforts have focused on linear regression and its variants (Bhatia et al., 2015; 2017; Shen & Sanghavi, 2019) or convex problems (Prasad et al., 2018). Thus, their results cannot be directly generalized to DNNs. Although Diakonikolas et al. (2019) presented a generalized non-convex optimization method with an agnostic corruption guarantee, the space complexity of the algorithm is high, and thus, cannot be applied to DNNs with large number of parameters. We will discuss Diakonikolas et al. (2019) in the next section.

3. Methodology

Before introducing our algorithm, we first present our corrupted supervision setting. To characterize agnostic corruptions, we assume there is an *adversary* that tries to corrupt the target variable of clean data. There is no restriction on how the adversary corrupts the supervision, which can either be randomly permuting the target, or in a way that maximizes its negative impact on the model performance. The adversary can choose up to ϵ fraction of the clean target $\mathbf{D}_y \in \mathbb{R}^{n\times q}$ and alters the selected rows of $\mathbf{D}_y$ to arbitrary valid numbers, generating $\mathbf{D}_y^\epsilon \in \mathbb{R}^{n\times q}$. The adversary then returns the corrupted dataset $\mathbf{D}_x, \mathbf{D}_y^\epsilon$ to our learning algorithm $\mathcal{A}$. The adversary can have full knowledge of the data or even the learning algorithm $\mathcal{A}$. The only constraint on the adversary is the corruption rate, ϵ. A key question is: *Given a dataset* $\mathbf{D}_x \in \mathbb{R}^{n\times p}$, $\mathbf{D}_y^\epsilon \in \mathbb{R}^{n\times q}$, *with* ϵ-

fraction of corrupted supervision, and a learning objective $\phi : \mathbb{R}^p \times \mathbb{R}^q \times \mathbb{R}^d \to \mathbb{R}$ *parameterized by* $\theta \in \mathbb{R}^d$*, can we output the parameters* θ *such that* $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$ *is minimized*?

When $\epsilon = 0$, $\mathbf{D}_y^\epsilon = \mathbf{D}_y$ and the learning is performed on clean data. The stochastic gradient descent algorithm may converge to a stationary point where $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\| = 0$. However, this is no longer the case when the supervision is corrupted as above due to the impact of the corrupted data on θ. We thus want an efficient algorithm to find a model θ that minimizes $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$. A robust model θ should have a small value of $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$, and we hypothesize that a smaller $\|\nabla_\theta \phi(\theta; \mathbf{D}_x, \mathbf{D}_y)\|$ leads to better generalization.

3.1. Stochastic Gradient Descent with Biased Gradient

A direct consequence of corrupted supervision is biased gradient estimation. In this section, we will first analyze how such biased gradient estimation affects the robustness of learning. The classical analysis of stochastic gradient descent (SGD) requires access to the stochastic gradient oracle, which is an unbiased estimation of the true gradient. However, corrupted supervision leads to corrupted gradients, which makes it difficult to get unbiased gradient estimation without assumptions on how the gradients are corrupted.

The convergence of biased gradient has been studied via a series of previous works (Schmidt et al., 2011; Bernstein et al., 2018; Hu et al., 2020; Ajalloeian & Stich, 2020; Scaman & Malherbe, 2020). For the sake of completeness, we use the informal theorem below to show how biased gradients affect the final convergence of SGD, and present a more formal version and its proof in the appendix.

Theorem 1 (Convergence of Biased SGD (Informal)) *Under mild assumptions, denote* ζ *as the maximum* ℓ_2 *norm of the difference between the clean mini-batch gradient and corrupted mini-batch gradient, i.e.,* $\|\mathbf{g} - \tilde{\mathbf{g}}\| \leq \zeta$*. By using* ζ*-biased gradient , SGD converges to the* ζ*-approximated stationary points:* $\mathbb{E}\left(\|\nabla \phi(\theta_t)\|^2\right) = \mathcal{O}(\zeta^2)$.

The difference between the above theorem and the typical convergence theorem for SGD is that we are using a biased gradient estimation. According to Theorem 1, robust estimation of the gradient $\mathbf{g}$ is the key to ensure a robust model that converges to the clean solution. We also assume the loss function has the form of $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$, in which many commonly used loss functions belong to this category.

3.2. Robust Gradient Estimation for General Data Corruption

Before discussing the corrupted supervision setting, we first review the general corruption setting, where the corruptions may be present in both the supervision and input features. A naïve approach is to apply a robust coordinate-wise gradient estimation approach such as coordinate-wise median for gradient estimation. However, by using the coordinate-wise robust estimator, the L2 norm of the difference between the estimated and ground-truth gradients contains a factor of $\mathcal{O}(\sqrt{d})$, where d is the gradient dimension. This error term induces a high penalty for high dimensional models and thus cannot be applied to DNNs. Recently, Diakonikolas et al. (2016) proposed a robust mean estimator with *dimension-free* error for general types of corruptions. Diakonikolas et al. (2019) achieves an error rate of $\mathcal{O}(\sqrt{\epsilon})$ for general corruption. This begs the question whether it is possible to further improve the $\mathcal{O}(\sqrt{\epsilon})$ error rate if we consider only corrupted supervision.

To motivate our main algorithm (Alg. 2), we first introduce and investigate Alg. 1 for general corruption with dimension-dependent error. The algorithm excludes data points with large gradient norms and uses the empirical mean of the remaining points to update the gradient. Cor. 1 below describes its robustness property.

Algorithm 1 (***PRL(G)***) Provable Robust Learning for General Corrupted Data

input: Label corrupted dataset $\mathbf{D}_x, \mathbf{D}_y^\epsilon$, learning rate γ_t;
return: model parameter θ;
for $t = 1$ to maxiter **do**
  Calculate the individual gradient $\tilde{\mathbf{G}}$ for sampled minibatch $\mathbf{M}$
  For each row $\mathbf{z}_i$ in $\tilde{\mathbf{G}}$, calculate the l2 norm $\|\mathbf{z}_i\|$
  Choose the ϵ-fraction rows with large $\|\mathbf{z}_i\|$
  Remove those selected rows, and return the empirical mean of the rest points as $\hat{\mu}$.
  Update model $\theta_{t+1} = \theta_t - \gamma_t \hat{\mu}$
end for

Corollary 1 (Robust Optimization For Corrupted Data) *Given the assumptions in Theorem 1, applying Algorithm 1 to* ϵ*-fraction corrupted data yields* $\min_{t \in [T]} \mathbb{E}\left(\|\nabla \phi(\theta_t)\|\right) = \mathcal{O}(\epsilon\sqrt{d})$ *for large enough iteration* T*, where* d *is the number of the parameters.*

Remark 1 *The term* $\sqrt{d}$ *is due to the upper bound of* d*-dimensional gradient norm of clean data. The term can be removed if we assume the gradient norm is uniformly bounded by* L*. However, this assumption is too strong for robust gradient estimation. We will show later that the assumption can be relaxed (i.e. bounded maximum singular value of gradient) under the corrupted supervision setting.*

The error bound in the above corollary has several practical issues. First, the bound grows with increasing dimensionality, and thus, is prohibitive when working with DNNs, which have extremely large gradient dimensions due to their massive number of parameters. Even though Cor. 1 can improve the factor $\sqrt{\epsilon}$ (Diakonikolas et al., 2019) to ϵ, the results remain impractical compared to the dimension-free

$\mathcal{O}(\sqrt{\epsilon})$ guarantee in (Diakonikolas et al., 2019), since above bound involves the dimension related term $\sqrt{d}$.

Efficiency is another main limitation of Alg. 1 since it requires computing individual gradients. Although there are advanced methods available to obtain the individual gradient, e.g., (Goodfellow, 2015), they are still relatively slow compared to the commonly used back-propagation algorithm. Moreover, many of them are not compatible with other components of DNN such as batch normalization (BN). Since the individual gradients are not independent within the BN, they will lose the benefits of parallelization. We will show below that the above issues can be addressed under the corrupted supervision setting and propose a practical solution that easily scales for DNNs.

3.3. Robust Gradient Estimation for One Dimensional Corrupted Supervision

In this section, we show that the robustness bound in Cor. 1 can be improved if we assume the corruption comes from the supervision only. In addition, by fully exploiting the gradient structure of the corrupted supervision, our algorithm is much more efficient and is compatible with batch normalization. We begin with a 1-dimensional supervision setting (e.g., binary classification or single-target regression) to illustrate this intuition and will extend it more general settings in the next section. Consider a supervised learning problem with input features $\mathbf{X} \in \mathbb{R}^{n\times p}$ and supervision $\mathbf{y} \in \mathbb{R}^n$. The goal is to learn a function f, parameterized by $\theta \in \mathbb{R}^d$, by minimizing the following loss $\min_\theta \sum_{i=1}^n \phi_i = \min_\theta \sum_{i=1}^n \mathcal{L}(y_i, f(\mathbf{x}_i, \theta))$. The gradient for a data point i is $\nabla_\theta \phi_i = \frac{\partial l_i}{\partial f_i}\frac{\partial f_i}{\partial \theta} = \alpha_i \mathbf{g}_i$.

In general, if the corrupted gradients drive the gradient estimation away from the clean gradient, they are either large in magnitude or systematically change the direction of the gradient (Diakonikolas et al., 2019). However, our key observation is that, when only the supervision is corrupted, the corruption contributes only to the term $\alpha_i = \frac{\partial l_i}{\partial f_i}$, which is a scalar in the one-dimensional setting. In other words, given the clean gradient of i^{th} point, $g_i \in \mathbb{R}^d$, the corrupted supervision only re-scales the gradient vector, changing the gradient from $\alpha_i \mathbf{g}_i$ to $\delta_i \mathbf{g}_i$, where $\delta_i = \frac{\partial l_i^\epsilon}{\partial f_i}$. As such, it is unlikely for the corrupted supervision to systematically change the gradient direction.

The fact that corrupted supervision re-scales the clean gradient can be exploited to reshape the robust optimization problem. Suppose we update our model in each iteration by $\theta^+ = \theta - \gamma\mu(\mathbf{G})$, where $\mu(\cdot)$ denotes the empirical mean function and $\mathbf{G} = [\nabla_\theta \phi_1^T, \ldots, \nabla_\theta \phi_m^T] \in \mathbb{R}^{m\times d}$ is the gradient matrix for a mini-batch of size m. We consider the following problem:

Problem 1 (Robust Gradient Estimation for One Dimensional Corrupted Supervision) *Given a clean gradient matrix $\mathbf{G} \in \mathbb{R}^{m\times d}$, an ϵ-corrupted matrix $\tilde{\mathbf{G}}$ with at most ϵ-fraction rows are corrupted from $\alpha_i \mathbf{g}_i$ to $\delta_i \mathbf{g}_i$, design an algorithm $\mathcal{A} : \mathbb{R}^{m\times d} \to \mathbb{R}^d$ that minimizes $\|\mu(\mathbf{G}) - \mathcal{A}(\tilde{\mathbf{G}})\|$.*

Note that when $\|\delta_i\|$ is large, the corrupted gradient will have a large effect on the empirical mean, and if $\|\delta_i\|$ is small, the corrupted gradient will have a limited effect on the empirical mean. This motivates us to develop an algorithm that filters out data points by the loss layer gradient $\|\frac{\partial l_i}{\partial f_i}\|$. If the norm of the loss layer gradient of a data point is large (in one-dimensional case, this gradient reduces to a scalar and the norm becomes its absolute value), we exclude the data point when computing the empirical mean of gradients for this iteration. Note that this algorithm is applicable to both regression and classification problems. In particular, for the mean squared error (MSE) loss used in regression, its gradient norm is exactly the loss itself, and the algorithm reduces to self-paced learning or trim loss (Shen & Sanghavi, 2019). We summarize the procedure in Algorithm 2 and will extend it to the more general multi-dimensional case in the next section.

Algorithm 2 (***PRL(L)***) Efficient Provable Robust Learning for Corrupted Supervision

input: dataset $\mathbf{D}_x, \mathbf{D}_y^\epsilon$ with corrupted supervision, learning rate γ_t;
return: model parameter θ;
for $t = 1$ to maxiter **do**
  Randomly sample a mini-batch $\mathbf{M}$ from $\mathbf{D}_x, \mathbf{D}_y^\epsilon$
  Compute the predicted label $\hat{\mathbf{Y}}$ from $\mathbf{M}$
  Calculate the gradient norm for the loss layer, (e.g., $\|\hat{\mathbf{y}} - \mathbf{y}\|$ for mean square error or cross entropy)
  $\tilde{\mathbf{M}} \leftarrow \mathbf{M} - \mathbf{M}_\tau$, where $\mathbf{M}_\tau$ is the top-τ fraction of data points with largest $\|\hat{\mathbf{y}} - \mathbf{y}\|$
  Update model $\theta_{t+1} = \theta_t - \gamma_t \hat{\mu}$, where $\hat{\mu}$ is the empirical mean gradient of $\tilde{\mathbf{M}}$
end for

3.4. Extension to Multi-Dimensional Corrupted Supervision

To extend our approach to multi-dimensional case, let q be the output dimension of y. The gradient for each data point i is $\nabla_\theta \phi_i = \frac{\partial l_i}{\partial f_i}\frac{\partial f_i}{\partial \theta}$, where $\frac{\partial l_i}{\partial f_i} \in \mathbb{R}^q$ is the gradient of the loss with respect to model output, and $\frac{\partial f_i}{\partial \theta} \in \mathbb{R}^{q\times d}$ is the gradient of the model output with respect to model parameters. When the supervision is corrupted, the corruption affects the term $\frac{\partial l_i}{\partial f_i}$, which is now a vector. Let $\delta_i = \frac{\partial l_i^\epsilon}{\partial f_i} \in \mathbb{R}^q$, $\alpha_i = \frac{\partial l_i}{\partial f_i} \in \mathbb{R}^q$, $\mathbf{W}_i = \frac{\partial f_i}{\partial \theta} \in \mathbb{R}^{q\times d}$, and m be the mini-batch size. Denote the clean gradient matrix as $\mathbf{G} \in \mathbb{R}^{m\times d}$, where the i_{th} row of gradient matrix $\mathbf{g}_i = \alpha_i \mathbf{W}_i$. The multi-dimensional robust gradient estimation problem is defined as follows.

Problem 2 (Robust Gradient Estimation for Multi-Dimensional Corrupted Supervision) *Given a clean gradient matrix* $\mathbf{G}$*, an* ϵ*-corrupted matrix* $\tilde{\mathbf{G}}$ *with at most* ϵ*-fraction rows corrupted from* $\alpha_i \mathbf{W}_i$ *to* $\delta_i \mathbf{W}_i$*, design an algorithm* $\mathcal{A} : \mathbb{R}^{m\times d} \to \mathbb{R}^d$ *that minimizes* $\|\mu(\mathbf{G}) - \mathcal{A}(\tilde{\mathbf{G}})\|$.

We begin our analysis by examining the effect of ***randomized filtering-based algorithms***, i.e., using the empirical mean gradient of randomly selected $(1-\epsilon)$-fraction of the data to estimate the clean averaged gradient. Randomized filtering-based algorithm is not a robust learning approach, but its analysis leads to important insights into designing one. We have the following lemma for any randomized filtering-based algorithm (proof is given in supplementary materials:

Lemma 1 (Gradient Estimation Error for Randomly Dropping ϵ-fraction Data) *Let* $\tilde{\mathbf{G}} \in \mathbb{R}^{m\times d}$ *be a corrupted matrix generated as in Problem 2 and* $\mathbf{G} \in \mathbb{R}^{m\times d}$ *be the original, clean gradient matrix. Suppose an arbitrary* $(1-\epsilon)$*-fraction rows are selected from* $\tilde{\mathbf{G}}$ *to form the matrix* $\mathbf{N} \in \mathbb{R}^{n\times d}$*. Let* μ *be the empirical mean function. Assume the clean gradient before loss layer has a bounded operator norm, i.e.,* $\|\mathbf{W}\|_{op} \leq C$*, the maximum clean gradient in loss layer* $\max_{i\in\mathbf{G}} \|\alpha_i\| = k$*, and the maximum corrupted gradient in loss layer* $\max_{i\in\mathbf{N}} \|\delta_i\| = v$*, then we have:*

$$\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| \leq Ck\frac{3\epsilon - 4\epsilon^2}{1-\epsilon} + Cv\frac{\epsilon}{1-\epsilon}.$$

Lemma 1 explains the factors affecting the robustness of filtering-based algorithm. Note that v is the only term that is related to the corrupted supervision. If v is large, the right-hand side can be arbitrarily large since an adversary can change the supervision in such a way that v becomes extremely large. Thus controlling the magnitude of v provides a way to effectively reduce the bound. For example, if we manage to control $v \leq k$, then the right-hand side cannot be too large. This can be achieved by sorting the gradient norms at the loss layer, and then discarding those belonging to the largest ϵ-fraction data points. Motivated by Lemma 1, we proposed Alg. 2, whose robustness guarantee is given by Thm. 2 and Cor. 2.

Theorem 2 (Robust Gradient Estimation For Supervision Corruption) *Let* $\tilde{\mathbf{G}}$ *be a corrupted matrix generated as in Problem 2,* q *be the output dimension, and* μ *be the empirical mean of the clean gradient matrix* $\mathbf{G}$*. Assuming the maximum clean gradient before loss layer has bounded operator norm:* $\|\mathbf{W}\|_{op} \leq C$*, then the output of gradient estimation in Algorithm 2,* $\hat{\mu}$*, satisfies* $\|\mu - \hat{\mu}\| = \mathcal{O}(\epsilon\sqrt{q}) \approx \mathcal{O}(\epsilon)$.

Thm. 2 can be obtained from Lemma 1 by substituting v by k. The following robustness guarantee can then be obtained by applying Thm. 1.

Corollary 2 (Robust Optimization For Corrupted Supervision Data) *Given the assumptions used in Thm. 1, applying Algorithm. 2 to any* ϵ*-fraction supervision corrupted data, yields* $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\mathbf{x}_t)\|\right) = \mathcal{O}(\epsilon\sqrt{q})$ *for large enough* T*, where* q *is the dimension of the supervision.*

Comparing Cor. 1 and Cor. 2, we see that when the corruption only comes from supervision, the dependence on d is reduced to q, where $q \ll d$ in most deep learning problems.

At first glance, the error bound $\mathcal{O}(\epsilon\sqrt{q})$ appears to suggest that learning a model for 10,000 classes with only 10% noise is more challenging than training a model for 100 classes with 80% noise. This statement seems to contradict existing results for many benchmark noisy-label tasks, and thus, needs to be clarified. First, the $\mathcal{O}(\epsilon\sqrt{q})$ bound is for any loss function, including both regression and classification. Second, the $\sqrt{q}$ term came from the norm of the q-dimensional loss layer gradient. For cross-entropy loss, its loss-layer gradient norm is $\|\hat{\mathbf{y}} - \mathbf{y}\|$. In classification, $\mathbf{y}$ is a one-hot encoding vector and $\hat{\mathbf{y}}$ is a probability vector. Thus, $\|\hat{\mathbf{y}} - \mathbf{y}\|$ is at most $\sqrt{2}$ and we can remove $\sqrt{q}$ out of the big-O notation for cross-entropy loss. This explains why we can achieve good results in classification tasks even when the target dimension is high. However, if it is regression or multi-task classification, then a higher target dimension increases the difficulty in achieving robust performance.

3.5. Comparison against Other Robust Optimization Methods

SEVER (Diakonikolas et al., 2019) provides state-of-the-art theoretical results for *general corruptions*, with a promising $\mathcal{O}(\sqrt{\epsilon})$ dimension-free guarantee. Compared to Diakonikolas et al. (2019), we have two contributions: **a)** When corruption comes only from the supervision, we show a better error rate if the supervision dimension can be treated as a small constant. **b)** Our algorithm can scale to DNNs unlike Diakonikolas et al. (2019), which is important as DNN models are currently state-of-the-art learning methods.

Despite the impressive theoretical results in Diakonikolas et al. (2019), it cannot be applied to DNNs even with the current best hardware configuration. Diakonikolas et al. (2019) used dimension-free robust mean estimation techniques to design the learning algorithm, while most robust mean estimation approaches rely on filtering data by computing the score of projection to the maximum singular vector. For example, the approach in Diakonikolas et al. (2019) requires applying expensive SVD on $n \times d$ individual gradient matrix, where n is the sample size and d is the number of parameters. This method works well for smaller datasets and smaller models when both n and d are small enough for current memory limitation. However, for DNNs, this matrix size is far beyond current GPU memory capability. For example, in our experiment, n is 60,000 and d is in the

order of millions (network parameters). It is impractical to store 60,000 copies of networks in a single GPU card. In contrast, our algorithm does not need to store the full gradient matrix. By only considering the loss-layer gradient norm, it can be easily extended to DNNs, and we show that this simple strategy works well in theory and challenging empirical tasks. We note that better robustness guarantee can be achieved in linear (Bhatia et al., 2015; 2017) or convex (Prasad et al., 2018) cases, but they cannot be directly applied to DNNs.

The strongest assumption behind our proof is that the maximum singular value of the gradient before loss layer is bounded. We also treat the clean gradient loss layer norm (k in Lemma 1) as a constant, which is particularly true for DNNs due to their overparameterization. In practice, our algorithm slowly increases the dropping ratio τ at first few epochs, which guarantees that k is a small number.

3.6. Relationship to Self-Paced Learning (SPL)

Many state-of-the-art methods with noisy labels depend on SPL (Han et al., 2018; Song et al., 2019; Yu et al., 2019; Shen & Sanghavi, 2019; Wei et al., 2020; Sun et al., 2020; Kolesnikov et al., 2019; Liang et al., 2016; Jiang et al., 2015; Meng et al., 2017; Fan et al., 2017; Jiang et al., 2014; 2020). At first glance, our method looks very similar to SPL. Instead of keeping data points with small gradient norms, SPL tries to keep those points with small loss. The gradient norm and loss function are related via the famous Polyak-Łojasiewicz (PL) condition. The PL condition assumes there exists some constant $s > 0$ such that $\forall \mathbf{x} : \frac{1}{2}\|\nabla\phi(\mathbf{x})\|^2 \geq s\,(\phi(\mathbf{x}) - \phi^*)$. As we can see, when the neural network is highly over-parameterized, ϕ^* can be assumed to be equal across different samples since the neural networks can achieve zero training loss (Zhang et al., 2016). By sorting the error $\phi(\mathbf{x}_i)$ for every data point, SPL is actually sorting the lower bound of the gradient norm if the PL condition holds. However, the ranking of gradient norm and the ranking of the loss can be very different since there is no guarantee that the gradient norm is monotonically increasing with the loss value. We provide an illustration as to why SPL is not robust from a geometric perspective in the supplementary materials. Here we show that the monotonic relationship can be easily violated even for the simple square loss function. One easy counter-example is $\phi(x_1, x_2) = 0.5x_1^2 + 50x_2^2$. Take two points (1000, 1) and (495, -49.5), we will find the monotonic relationship does not hold for these two points. Nocedal et al. (2002) showed that the monotonic relationship holds for ***square loss*** (i.e. $\phi(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\mathbf{Q}(\mathbf{x}-\mathbf{x}^*)$) if the condition number of Q is smaller than $3 + 2\sqrt{2}$, which is quite a strong assumption especially when $\mathbf{x}$ is in high-dimension. Thus, although SPL sorts the lower bound of the gradient norm under mild assumptions, our algorithm is significantly different from SPL and its variations.

Next, we discuss the relationship between SPL and Algorithm 2 under corrupted supervision. SPL has the same form as Algorithm 2 when we are using mean square error to perform regression tasks since the loss layer gradient norm is equal to loss itself. However, in classification, Algorithm 2 is different from the SPL. In order to better understand the algorithm, we further analyze the difference between SPL and our algorithm for cross-entropy loss.

For cross entropy, denote the output logit as $\mathbf{o}$, we have $H(\mathbf{y}_i, \mathbf{f}_i) = -\langle \mathbf{y}_i, \log(\text{softmax}(\mathbf{o}_i))\rangle = -\langle \mathbf{y}_i, \log(\mathbf{f}_i)\rangle$. The gradient norm of cross entropy with respect to $\mathbf{o}_i$ is: $\frac{\partial H_i}{\partial \mathbf{o}_i} = \mathbf{y}_i - \text{softmax}(\mathbf{o}_i) = \mathbf{f}_i - \mathbf{y}_i$. Thus, the gradient of loss layer is the MSE between $\mathbf{y}_i$ and $\mathbf{f}_i$. Next, we investigate when MSE and cross entropy has a non-monotonic relationship. For simplicity, we only consider the sufficient condition for the non-monotonic relationship, which is given by Lemma 2.

Lemma 2 *Let $\mathbf{y} \in \mathbb{R}^q$, where $\mathbf{y}_k = 1$ and $\mathbf{y}_i = 0$ for $i \neq k$. Suppose α and β are two q-dimensional vectors in probability simplex. Without loss of generality, suppose α has a smaller cross entropy loss and $\alpha_k \geq \beta_k$, then the sufficient condition for $\|\alpha - \mathbf{y}\| \geq \|\beta - \mathbf{y}\|$ is $\text{Var}_{i\neq k}(\{\alpha_i\}) - \text{Var}_{i\neq k}(\{\beta_i\}) \geq \frac{q}{(q-1)^2}\left((\alpha_k - \beta_k)(2 - \alpha_k - \beta_k)\right)$*

As $\alpha_k \geq \beta_k$, the term on right-hand-side of the inequality is non-negative. Thus, when MSE generates a result that differs from cross-entropy, the variance in the probability vector of the non-true class for the discarded data point is larger. For example, consider the ground-truth vector $\mathbf{y} = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$, and two prediction vectors, $\alpha = [0.08, 0.28, 0.08, 0.08, 0.08, 0.08, 0.08, 0.08, 0.08, 0.08]$ and $\beta = [0.1, 0.3, 0.34, 0.05, 0.05, 0.1, 0.03, 0.03, 0, 0]$. α has a smaller MSE loss while β has a smaller cross-entropy loss. β will more likely be noisy data since it has two relatively large values of 0.3 and 0.34. Since cross entropy loss considers only one dimension, corresponding to the ground truth label, it cannot detect such a situation. Compared to cross-entropy, the gradient (mse loss) considers all dimensions, and thus, will consider the distribution of the overall prediction.

3.7. Integration with Co-teaching Style Training

Co-teaching (Han et al., 2018) is one of the state-of-the-art deep methods for learning with noisy labels. Motivated by Co-teaching, we propose ***Co-PRL(L)***, which has the same framework as co-teaching but uses the loss-layer gradient to select the data. The full algorithm is shown in the supplementary materials. The key difference between ***Co-PRL(L)*** and algorithm 2 is that in ***Co-PRL(L)***, we optimize two network by ***PRL(L)***. Also in every iteration, two networks will exchange the selected data to update their own parameters.

4. Experimental Results

We have performed our experiments on various benchmark regression and classification datasets. We compare ***PRL(G)*** (Algorithm 1), ***PRL(L)*** (Algorithm 2), and ***Co-PRL(L)*** (Algorithm 4 in supplementary materials) against the following baselines. ***Standard***: standard training without filtering data (mse for regression, cross entropy for classification); ***Normclip***: training with norm clipping; ***Huber***: training with huber loss (for regression only); ***Decouple*** (Malach & Shalev-Shwartz, 2017): decoupling network, update two networks by using their disagreement (for classification only); ***Bootstrap*** (Reed et al., 2014): uses a weighted combination of predicted and original labels as the correct labels, and then perform back propagation (for classification only); ***Min-sgd*** (Shah et al., 2020): chooses the smallest loss sample in minibatch to update model; ***SPL*** (Jiang et al., 2018): self-paced learning (also known as the trimmed loss or predefined curriculum) by dropping the data points with large losses (same as ***PRL(L)*** in regression setting with MSE loss); ***Ignormclip***: clipping individual gradients and then average them to update model (regression only); ***Co-teaching*** (Han et al., 2018): collaboratively train a pair of SPL models and exchange their selected data to another model (for classification only).

Since it is hard to design experiments for ***agnostic corrupted supervision***, we analyzed the performance on a broad class of corrupted supervision settings: ***linadv***: corrupted supervision is generated from a random linear model: $\mathbf{Y}_\epsilon = \mathbf{X} * \mathbf{W}_\epsilon$ (regression); ***signflip***: corrupted supervision is obtained by $\mathbf{Y}_\epsilon = -\mathbf{Y}$ (regression); ***uninoise***: corrupted supervision is a random sample from uniform distribution, $\mathbf{Y}_\epsilon \sim [-5, 5]$ (regression); ***mixture***: mixture of above types of corruptions (regression); ***pairflip***: shuffle the target values (e.g., coordinates for eyes to those for mouth in CelebA or cat to dog in CIFAR) (regression and classification); ***symmetric***: randomly assign wrong class label (classification). We use accuracy as the evaluation metric for classification and R-square for regression experiments. Due to space limitation, we only show the average evaluation score on testing data for the last 10 epochs. We also include part of the training curves to show how the test evaluation metric changes during the training phase. ***The whole training curves are provided in the supplementary materials.*** Note that the regression experiments are repeated 5 times while the classification experiments are repeated 3 times.

4.1. Regression Results

For regression, we evaluated our method on the CelebA dataset, which contains 162,770 training images, 19,867 validation images, and 19,962 test images. Given a human face image, the goal is to predict the coordinates for 10 landmarks in the face image. Specifically, the target variable is a ten-dimensional vector of coordinates for the left eye, right eye, nose, left mouth, and right mouth. We added different types of corruption to the landmark coordinates. The CelebA dataset is preprocessed as follows: we use a three-layer CNN to train 162770 training images to predict clean coordinates (we use 19867 validation images to do the early stopping). We then apply the network to extract a 512-dimensional feature vector from the testing data. Thus, the final dataset after preprocessing consists of the feature sets $\mathbf{X} \in \mathbb{R}^{19962\times 512}$ and the target variable $\mathbf{Y} \in \mathbb{R}^{19962\times 10}$. We further split the data into 80% training and 20% test sets. We then manually add the ***linadv***, ***signflip***, ***uninoise***, ***pairflip***, and ***mixture*** corruptions to the target variable in the training set. For each type of corruption, the corruption rate is varied from 0.1 to 0.4. We use a 3-layer fully connected network for our experiments. The averaged r-square for the last 10 epochs are shown in Table 1. The training curves could be found in the second row of Figure 1. We can see Co-PRL(L) performs best in most cases. Surprisingly, the performance of PRL(G) is comparable to PRL(L). This is partially due to the shallow network structure and initialization. Another possible reason is that for this task, the gradient norm is upper bounded by a small constant.

4.2. Classification Results

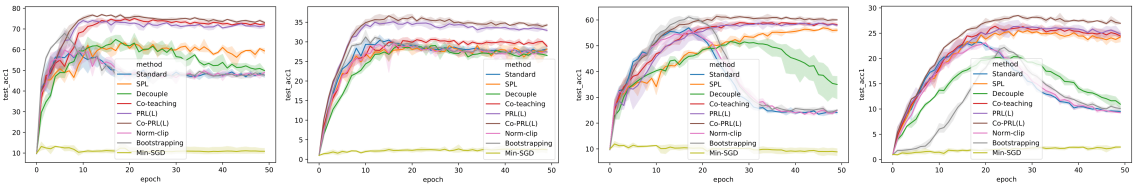
We perform our experiments on the CIFAR10 and CIFAR100 datasets to illustrate the effectiveness of our algorithm in classification setting. We use a 9-layer convolutional neural network, similar to the approach used in (Han et al., 2018). Since most baselines include batch normalization, it is difficult to get individual gradient efficiently, we exclude the ignormclip and PRL baselines. In the appendix, we attached the results if both co-teaching and Co-PRL(L) excludes the batch normalization module. Our results suggest that co-teaching cannot maintain robustness unlike our proposed method. Also, to compare against the current state of the art method, for ***symmetric*** noise, we use a corruption rate higher than 0.5. Although our theoretical analysis assumes the rate is less than 0.5, we empirically show that our method can also deal with higher corruption rates. The results for CIFAR10 and CIFAR100 are shown in Table 2. The results suggest that our method performs significantly better than the baselines irrespective of whether we are using one network (PRL vs SPL) or two networks (Co-PRL(L) vs Co-teaching). The training curves are given in the first row of Figure 1. Since the true corruption rate in real-world data is often unknown, we perform sensitivity analysis to show the effect of overestimating and underestimating ϵ in classification tasks. Based on the results in Table 3, we observe that overestimating ϵ leads to better performance in most cases because the corruption rate may vary in each mini-batch. Thus, overestimating ϵ can guarantee that the dangerous corrupted data points will be dropped.

Corruption	Standard	Normclip	Huber	Min-sgd	Ignormclip	PRL(G)	PRL(L)	Co-PRL(L)
linadv: 10	-2.33±0.84	-2.22±0.74	0.868±0.01	0.103±0.03	0.68±0.07	**0.876±0.01**	**0.876±0.01**	**0.876±0.01**
linadv: 20	-8.65±2.1	-8.55±2.2	0.817±0.015	0.120±0.02	0.367±0.28	**0.871±0.01**	0.869±0.01	0.869±0.01
linadv: 30	-18.529±4.04	-19.185±4.31	0.592±0.07	0.146±0.03	-0.944±0.51	**0.865±0.01**	0.861±0.01	0.860±0.01
linadv: 40	-32.22±6.32	-32.75±7.07	-2.529±1.22	0.180±0.01	-1.60 ± 0.80	**0.857± 0.01**	0.847±0.02	0.847±0.02
signflip: 10	0.800±0.02	0.798±0.03	0.857±0.01	0.110±0.04	0.846±0.01	0.877±0.01	0.878±0.01	**0.879±0.01**
signflip: 20	0.641±0.05	0.638±0.04	0.786±0.02	0.105±0.07	0.82±0.02	0.875±0.01	0.875±0.01	**0.877±0.01**
signflip: 30	0.422±0.04	0.421±0.04	0.629±0.03	0.124±0.05	0.795±0.02	0.871±0.01	0.873±0.01	**0.875±0.01**
signflip: 40	0.193±0.043	0.190±0.04	0.379±0.05	-0.028±0.25	0.759±0.01	**0.872±0.01**	**0.872±0.01**	0.871±0.01
uninoise: 10	0.845±0.01	0.844±0.01	0.875±0.01	0.103±0.03	0.859±0.01	0.879±0.01	**0.881±0.01**	**0.881±0.01**
uninoise: 20	0.798±0.02	0.795±0.02	0.865±0.01	0.120±0.02	0.844±0.01	0.878±0.01	**0.880±0.01**	**0.880±0.01**
uninoise: 30	0.728±0.02	0.725±0.02	0.847±0.01	0.146±0.03	0.831±0.01	0.878±0.01	**0.879±0.01**	**0.879±0.01**
uninoise: 40	0.656±0.02	0.654±0.02	0.825±0.01	0.180±0.01	0.821±0.01	0.876± 0.01	**0.878±0.01**	**0.878±0.01**
pairflip: 10	0.852±0.02	0.851±0.02	0.870±0.01	0.110±0.04	0.867±0.01	0.877±0.01	0.876±0.01	**0.878±0.01**
pairflip: 20	0.784±0.03	0.783±0.03	0.841±0.02	0.120±0.03	0.849±0.01	**0.874±0.01**	0.873±0.01	**0.874±0.01**
pairflip: 30	0.688±0.04	0.686±0.04	0.770±0.02	0.133±0.02	0.828±0.01	0.870±0.01	0.872±0.01	**0.873±0.01**
pairflip: 40	0.556±0.06	0.553±0.06	0.642±0.06	0.134±0.03	0.810±0.02	0.863±0.01	**0.870±0.01**	**0.870±0.01**
mixture: 10	-0.212±0.6	-0.010±0.48	0.873±0.01	0.101±0.03	0.861±0.01	0.878±0.01	**0.880±0.01**	**0.880±0.01**
mixture: 20	-0.404±0.68	-0.463±0.67	0.855±0.01	0.119±0.03	0.855±0.01	0.877±0.01	0.878±0.01	**0.879±0.01**
mixture: 30	-0.716±0.57	-0.824±0.39	0.823±0.01	0.148±0.02	0.847±0.01	0.875±0.01	0.877±0.01	**0.878±0.01**
mixture: 40	-3.130±1.51	-2.69±0.84	0.763±0.01	0.175±0.02	0.835±0.01	0.872±0.01	0.875 ±0.01	**0.876±0.01**

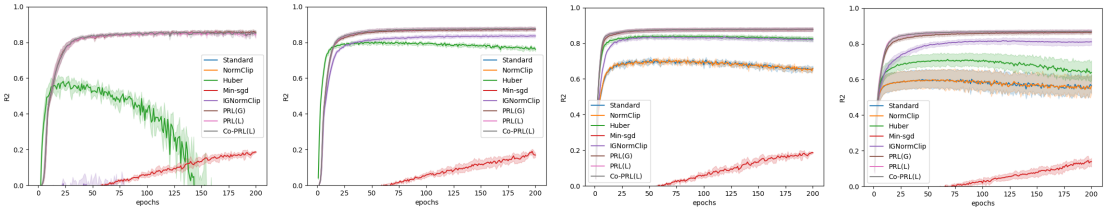
Table 1. R-square on CelebA clean testing data, and the standard deviation is from last ten epochs and 5 random seeds.

Corruption	Standard	Normclip	Bootstrap	Decouple	Min-sgd	SPL	PRL(L)	Co-teaching	Co-PRL(L)
CF10-sym-30	63.22±0.18	62.41±0.06	63.67±0.24	70.73±0.51	13.31±2.24	77.77±0.34	79.40±0.19	79.90±0.13	**80.05±0.12**
CF10-sym-50	44.63±0.18	43.99±0.28	46.13±0.18	57.48±1.98	13.33±2.85	72.22±0.15	74.17±0.15	74.25±0.41	**75.43±0.09**
CF10-sym-70	24.12±0.09	24.17±0.37	25.13±0.39	40.11±4.62	9.08±0.94	56.19±0.33	58.36±0.62	58.41±0.33	**60.26±0.42**
CF10-pf-25	68.34±0.30	67.92±0.43	68.71±0.32	75.59±0.35	10.45±0.60	75.79±0.44	80.54±0.07	80.18±0.21	**81.51±0.13**
CF10-pf-35	58.68±0.28	58.27±0.18	58.19±0.12	66.38±0.44	12.29±1.92	70.40±0.27	77.61±0.35	77.97±0.03	**79.01±0.14**
CF10-pf-45	48.05±0.25	48.03±0.54	47.84±0.32	51.54±0.81	10.94±1.28	58.95±0.59	71.42±0.24	72.43±0.31	**73.78±0.17**
CF100-sym-30	32.83±0.39	32.10±0.64	34.47±0.22	32.95±0.44	2.94±0.61	44.37±0.44	46.40±0.18	45.02±0.29	**47.51±0.47**
CF100-sym-50	20.47±0.44	19.73±0.29	21.59±0.44	21.02±0.36	2.35±0.45	37.89±0.16	38.38±0.65	38.79±0.33	**40.64±0.11**
CF100-sym-70	9.93±0.07	9.93±0.23	10.59±0.17	12.55±0.46	2.32±0.24	24.10±0.44	25.38±0.56	24.94±0.53	**27.27±0.01**
CF100-pf-25	40.37±0.55	39.34±0.35	40.22±0.37	39.43±0.27	2.62±0.26	40.48±0.72	47.57±0.37	42.97±0.10	**48.06±0.26**
CF100-pf-35	34.07±0.19	32.88±0.10	34.53±0.23	33.14±0.07	2.30±0.07	34.17±0.46	43.32±0.16	36.69±0.23	**44.08±0.33**
CF100-pf-45	27.66±0.50	27.35±0.61	27.56±0.23	26.83±0.41	2.55±0.52	27.55±0.66	33.31±0.10	29.71±0.20	**34.43±0.05**

Table 2. Classification accuracy for clean testing data on CIFAR10 and CIFAR100 with training on ***symmetric*** and ***pairflip*** label corruption. The standard deviation is from last ten epochs and 3 random seeds.



(a) CF10-pairflip-0.45 (b) CF100-pairflip-0.45 (c) CF10-sym-70 (d) CF100-sym-70



(e) CelebA-linadv-0.4 (f) CelebA-mixture-0.4 (g) CelebA-uninoise-0.4 (h) CelebA-pairflip-0.4

Figure 1. Testing accuracy/R-square for CIFAR10, CIFAR100, and CelebA during the training phase.

Data	$\epsilon - 0.1$	$\epsilon - 0.05$	ϵ	$\epsilon + 0.05$	$\epsilon + 0.1$
CF10-Pair-45%	65.07±0.83	70.07±0.67	73.78±0.17	77.56±0.55	79.36±0.43
CF10-Sym-50%	69.21±0.35	72.53±0.45	75.43 ± 0.09	77.65±0.27	78.10±0.31
CF10-Sym-70%	53.88±0.64	58.49±0.97	60.26 ± 0.42	60.89±0.43	54.91±0.68
CF100-Pair-45%	32.60±0.45	34.17±0.40	34.43 ± 0.05	36.87±0.41	38.34±0.78
CF100-Sym-50%	37.74±0.41	39.72±0.36	40.64 ± 0.11	43.02±0.36	43.92±0.61
CF100-Sym-70%	24.40±0.47	25.50±0.45	27.27 ± 0.10	27.80±0.50	28.20±0.97

Table 3. Sensitivity analysis for over-estimated/under-estimated ϵ.

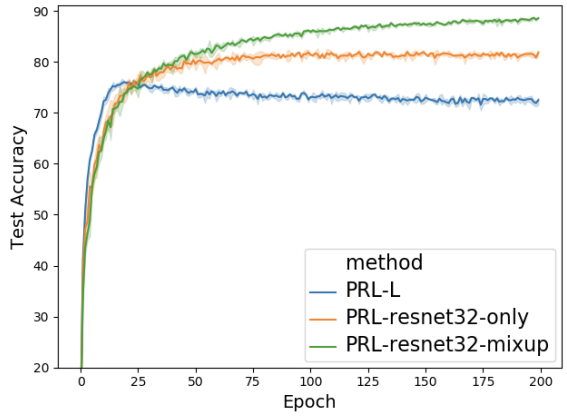


Figure 2. Improvement of PRL(L) under sym-50 label noise in CIFAR10 by using resnet32 and mixup for 3 random seeds.

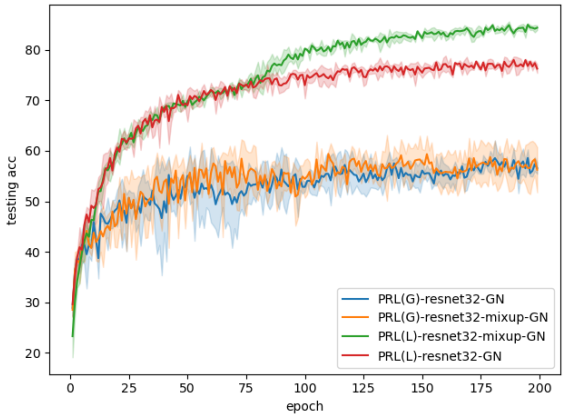


Figure 3. PRL(G) vs PRL(L) in resnet-32 under sym-50 label noise in CIFAR10 for 3 random seeds.

4.3. PRL(L) on Residual Network with Mixup

The previous experiments focused on evaluating the effect of the filtering step and did not consider its applicability to more advanced deep learning techniques to boost model performance. The simplicity of the PRL(L) framework allows it to be easily implemented by many other methods to boost model performance. To demonstrate this, we added several advanced components to PRL(L) and evaluate the performance improvement. Specifically, we added data augmentation, mix-up (Zhang et al., 2018) (i.e. using mixup on PRL(L) selected data), and a deeper network (resnet-32). We test the model performance on symmetric 50% label noise in CIFAR10, and the results are shown in Figure 2. The model performance indeed improves after changing the architecture from a 9-layer CNN to resnet-32. Furthermore, by adding the mixup component, the performance is boosted again, which is comparable to current state-of-the-art results.

4.4. Comparison between PRL(L) and PRL(G)

Our previous classification experiments did not include the results of PRL-G on CIFAR data since both 9-layer CNN and resnet-32 contain a batch normalization module, which is not compatible with PRL(G). Our theory suggested that PRL-L should outperform PRL-G when using a deeper network. To validate our theorem, we replace all batch normalization modules in resnet-32 with group normalization so that the individual gradients can be calculated efficiently. We test PRL(G) and PRL(L) on CIFAR10 with 50% symmetric label noise. As shown by the results given in Figure 3, PRL(L) significantly outperforms PRL(G) regardless of whether mixup is used. We also found that PRL(L) can be boosted by adding mixup data augmentation while mixup fails to improve PRL(G). These results further validate our theorem and show the superiority of PRL(L) compared to PRL(G).

5. Conclusion

In this paper, we proposed a simple yet effective algorithm to defend against agnostic supervision corruptions. Both the theoretical and empirical analysis showed the effectiveness of our algorithm. For future research, there are two questions that deserved further study. The first question is whether we can further improve $\mathcal{O}(\epsilon)$ error bound or show that $\mathcal{O}(\epsilon)$ is tight. The second question is how we can utilize more properties of neural networks, such as the sparse or low-rank structure in gradient to design better algorithms.

Acknowledgements

This research was supported in part by the grant National Science Foundation IIS-2006633, EF-1638679, IIS-1749940, Office of Naval Research N00014-20-1-2382, National Institute on Aging RF1AG072449. Any use of trade, firm or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

References

Ajalloeian, A. and Stich, S. U. Analysis of sgd with biased gradient estimators. *arXiv preprint arXiv:2008.00051*, 2020.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Bhatia, K., Jain, P., and Kar, P. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems*, pp. 721–729, 2015.

Bhatia, K., Jain, P., Kamalaruban, P., and Kar, P. Consistent robust regression. In *Advances in Neural Information Processing Systems*, pp. 2110–2119, 2017.

Cheng, Y., Diakonikolas, I., Ge, R., and Soltanolkotabi, M. High-dimensional robust mean estimation via gradient descent. *arXiv preprint arXiv:2005.01378*, 2020.

d'Aspremont, A. Smooth optimization with approximate gradient. *SIAM Journal on Optimization*, 19(3):1171–1183, 2008.

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Moitra, A., and Stewart, A. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 655–664. IEEE, 2016.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Steinhardt, J., and Stewart, A. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pp. 1596–1606, 2019.

Dong, Y., Hopkins, S., and Li, J. Quantum entropy scoring for fast robust mean estimation and improved outlier detection. In *Advances in Neural Information Processing Systems*, pp. 6067–6077, 2019.

Fan, Y., He, R., Liang, J., and Hu, B. Self-paced learning: an implicit regularization perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Goodfellow, I. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*, 2015.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pp. 8527–8537, 2018.

Hu, Y., Zhang, S., Chen, X., and He, N. Biased stochastic gradient descent for conditional stochastic optimization. *arXiv preprint arXiv:2002.10790*, 2020.

Huber, P. J. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pp. 492–518. Springer, 1992.

Jiang, L., Meng, D., Yu, S.-I., Lan, Z., Shan, S., and Hauptmann, A. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pp. 2078–2086, 2014.

Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. Self-paced curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pp. 2304–2313, 2018.

Jiang, L., Huang, D., Liu, M., and Yang, W. Beyond synthetic noise: Deep learning on controlled noisy labels. In *International Conference on Machine Learning*, pp. 4804–4815. PMLR, 2020.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6(2):8, 2019.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in neural information processing systems*, pp. 1189–1197, 2010.

Lai, K. A., Rao, A. B., and Vempala, S. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 665–674. IEEE, 2016.

Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, L.-J. Learning from noisy labels with distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1910–1918, 2017.

Liang, J., Jiang, L., Meng, D., and Hauptmann, A. G. Learning to detect concepts from webly-labeled video data. In *IJCAI*, volume 1, pp. 3–1, 2016.

Malach, E. and Shalev-Shwartz, S. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pp. 960–970, 2017.

Meng, D., Zhao, Q., and Jiang, L. A theoretical understanding of self-paced learning. *Information Sciences*, 414: 319–328, 2017.

Menon, A. K., Rawat, A. S., Reddi, S. J., and Kumar, S. Can gradient clipping mitigate label noise? In *International Conference on Learning Representations*, 2019.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. In *Advances in neural information processing systems*, pp. 1196–1204, 2013.

Nocedal, J., Sartenaer, A., and Zhu, C. On the behavior of the gradient norm in the steepest descent method. *Computational Optimization and Applications*, 22(1):5–35, 2002.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. 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.

Prasad, A., Suggala, A. S., Balakrishnan, S., and Ravikumar, P. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

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

Scaman, K. and Malherbe, C. Robustness analysis of non-convex stochastic gradient descent using biased expectations. *Advances in Neural Information Processing Systems*, 33, 2020.

Schmidt, M., Roux, N. L., and Bach, F. R. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*, pp. 1458–1466, 2011.

Shah, V., Wu, X., and Sanghavi, S. Choosing the sample with lowest loss makes sgd robust. In *International Conference on Artificial Intelligence and Statistics*, pp. 2120–2130. PMLR, 2020.

Shen, Y. and Sanghavi, S. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, pp. 5739–5748. PMLR, 2019.

Song, H., Kim, M., and Lee, J.-G. Selfie: Refurbishing unclean samples for robust deep learning. In *International Conference on Machine Learning*, pp. 5907–5915. PMLR, 2019.

Sun, M., Xing, J., Chen, B., and Zhou, J. Robust collaborative learning with noisy labels. In *2020 IEEE International Conference on Data Mining (ICDM)*, 2020.

Tukey, J. W. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, volume 2, pp. 523–531, 1975.

Wei, H., Feng, L., Chen, X., and An, B. Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13726–13735, 2020.

Yi, K. and Wu, J. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7017–7025, 2019.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pp. 7164–7173. PMLR, 2019.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Zheng, S., Wu, P., Goswami, A., Goswami, M., Metaxas, D., and Chen, C. Error-bounded correction of noisy labels. In *International Conference on Machine Learning*, 2020.

Supplementary Materials

1. Co-PRL(L) Algorithm

We borrow the framework from the co-teaching framework(Han et al., 2018). The only difference is the filtering criteria. Co-teaching uses loss value as the filtering criteria while Co-PRL(L) uses the loss-layer-gradient norm as the filtering criteria.

Algorithm 1 Co-PRL(L)

input: initialize w_f and w_g, learning rate η, fixed τ, epoch T_k and T_{max}, iterations N_{max}
Return: model parameter w_f and w_g
for $T = 1, 2, ..., T_{max}$ **do**
  for $N = 1, ..., N_{max}$ **do**
    random sample a minibatch $\mathbf{M}$ from $\mathbf{D}_x, \mathbf{D}_y^\epsilon$ (noisy dataset)
    get the predicted label $\hat{\mathbf{Y}}_f$ and $\hat{\mathbf{Y}}_g$ from $\mathbf{M}$ by w_f. w_g
    calculate the individual loss $l_f = \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_f)$, $l_g = \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_g)$
    calculate the gradient norm of loss layer $score_f = \|\frac{\partial l_f}{\partial \hat{\mathbf{y}}_f}\|$, $score_g = \|\frac{\partial l_g}{\partial \hat{\mathbf{y}}_g}\|$.
    sample $R(T)\%$ small-loss-layer-gradient-norm instances by $score_f$ and $score_g$ to get $\mathbf{N}_f$, $\mathbf{N}_g$
    update $w_f = w_f - \eta \nabla_{w_f} \mathcal{L}(\mathbf{N}_f, w_f)$, $w_g = w_g - \eta \nabla_{w_g} \mathcal{L}(\mathbf{N}_g, w_g)$ (selected dataset)
    update model $\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \hat{\mu}$
  end for
  Update $R(T) = 1 - \min\left\{\frac{T}{T_k}\tau, \tau\right\}$
end for

2. Further Illustration of the difference between SPL and PRL(G)

In this section, we will further illustrate the difference between SPL and PRL(G). In order to have a more intuitive understanding of our algorithm, we could look at the Figure 1(a) and 1(b). Since we are in the agnostic label corruption setting, it is difficult to filtering out the correct corrupted data. We showed two situations when loss filtering failed and gradient filtering failed. As we could see that when loss filtering method failed, the remaining corrupted data could have large impact on the overall loss surface while when gradient filtering method failed, the remaining corrupted data only have limited impact on the overall loss surface, thus gaining robustness.

3. Networks and Hyperparameters

The hyperparameters are in Table 1. For Classification, we use the same hyperparameters in (Han et al., 2018). For CelebA, we use 3-layer fully connected network with 256 hidden nodes in hidden layer and leakly-relu as activation function. We also released our code in `https://github.com/illidanlab/PRL`.

Data\HyperParameter	BatchSize	Learning Rate	Optimizer	Momentum
CF-10	128	0.001	Adam	0.9
CF-100	128	0.001	Adam	0.9
CelebA	512	0.0003	Adam	0.9

Table 1. Main Hyperparmeters

Data	$\epsilon - 0.1$	$\epsilon - 0.05$	ϵ	$\epsilon + 0.05$	$\epsilon + 0.1$
CF10-Pair-45%	65.07±0.83	70.07±0.67	73.78±0.17	77.56±0.55	79.36±0.43
CF10-Sym-50%	69.21±0.35	72.53±0.45	75.43 ± 0.09	77.65±0.27	78.10±0.31
CF10-Sym-70%	53.88±0.64	58.49±0.97	60.26 ± 0.42	60.89±0.43	54.91±0.68
CF100-Pair-45%	32.60±0.45	34.17±0.40	34.43 ± 0.05	36.87±0.41	38.34±0.78
CF100-Sym-50%	37.74±0.41	39.72±0.36	40.64 ± 0.11	43.02±0.36	43.92±0.61
CF100-Sym-70%	24.40±0.47	25.50±0.45	27.27 ± 0.10	27.80±0.50	28.20±0.97

Table 2. sensitivity analysis for estimated ϵ

4. Learning Curve

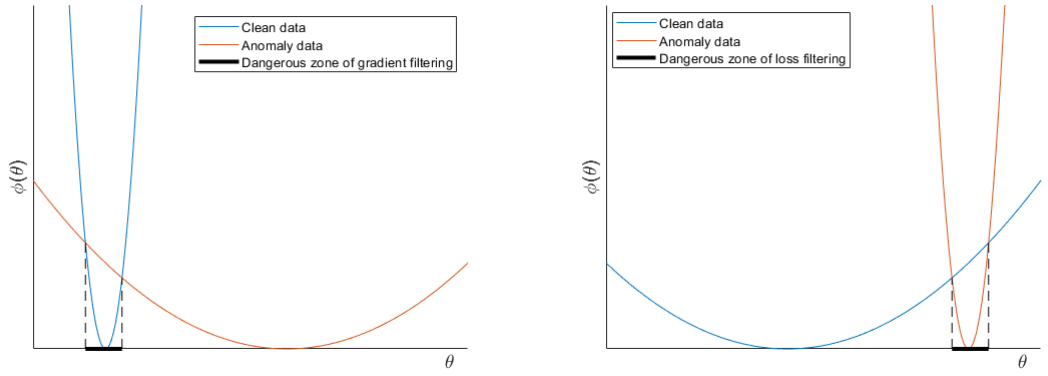
We show how testing evaluation changes along the training process for both classifcation and regression tasks in this section. The regression curve for CelebA data is showed in Figure 2. Note the for regression, the SPL and co-teaching are actually equivalent to our algorithm (i.e. PRL(L) and (Co-PRL(L))). The classification curve is in Figure 3.

5. Sensitivity Analysis

Since in real-world problems, it is hard to know that the ground-truth corruption rate, we perform the sensitivity analysis in classification tasks to show the effect of ϵ. The results are in Table 2. As we could see, the performance is stable if we overestimate the corruption rate, this is because only when we overestimate the ϵ, we could guarantee that the gradient norm of the remaining set is small. However, when we underestimate the corruption rate, in the worst case, there is no guarantee that the gradient norm of the remaining set is small. By using the empirical mean, even one large bad individual gradient would ruin the gradient estimation, and according to the convergence analysis of biased gradient descent, the final solution could be very bad in terms of clean data. That explains why to underestimate the corruption rate gives bad results. Also, from Table 2, we could see that using the ground truth corruption rate will lead to small uncertainty.

6. Empirical Results on Running Time

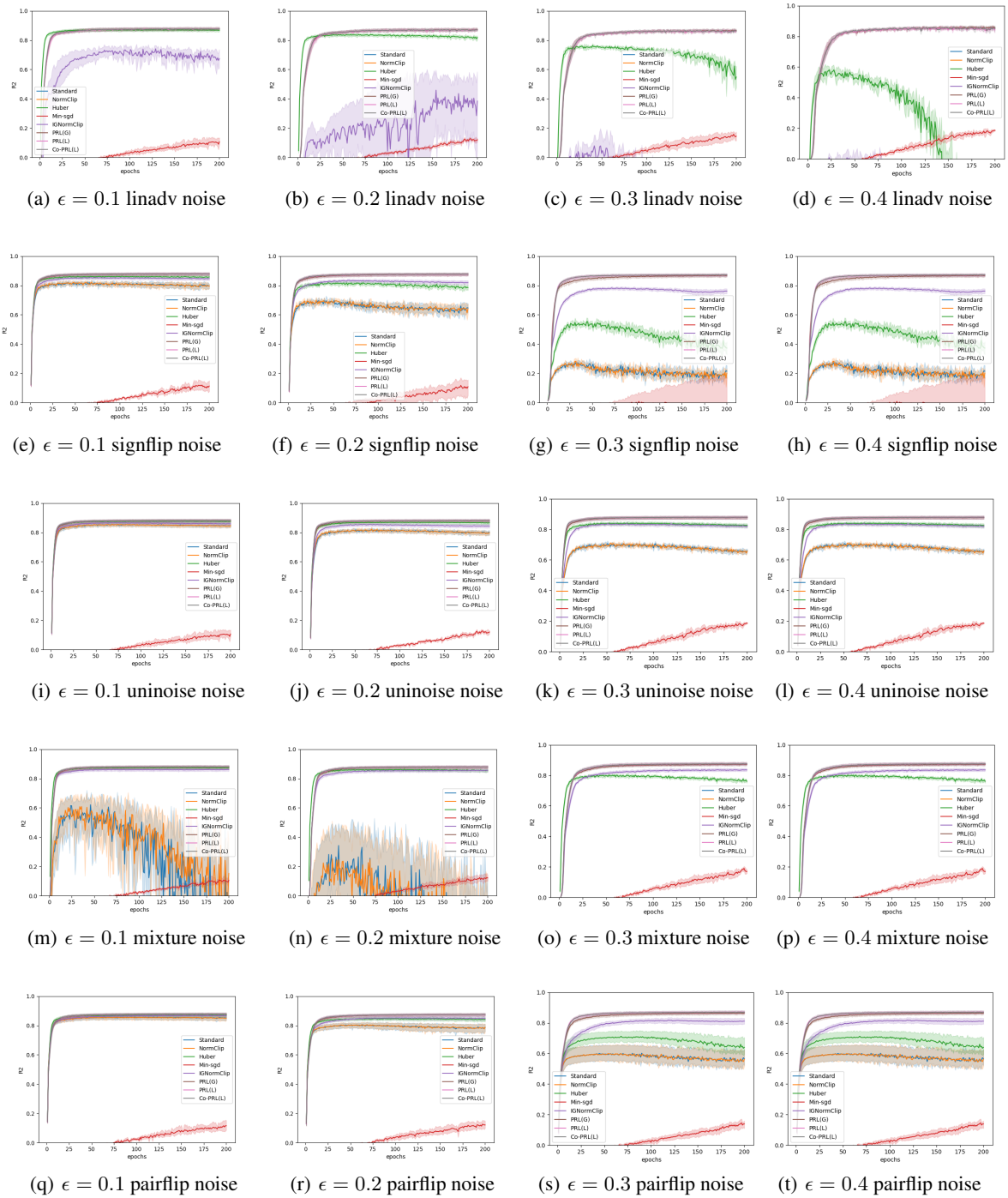
As we claimed in paper, the algorithm 2 (PRL(G)) is not efficient. In here we attached the execution time for one epoch for three different methods: ***Standard***, ***PRL(G)***, ***PRL(L)***. For fair comparison, we replace all batch normalization module to group normalization for this comparison, since it is hard to calculate individual gradient when using batch normalization. For PRL(G), we use opacus libarary to calculate the individual gradient. The results are showed in Table 3



(a) When gradient filtering method failed to pick out right corrupted data, the remaining corrupted data is relatively smooth, thus has limited impact on overall loss surface.

(b) When loss filtering method failed to pick out right corrupted data, the remaining corrupted data could be extremely sharp, thus has large impact on overall loss surface.

Figure 1. Further Illustration of difference between SPL and PRL(G)



(a) $\epsilon = 0.1$ linadv noise (b) $\epsilon = 0.2$ linadv noise (c) $\epsilon = 0.3$ linadv noise (d) $\epsilon = 0.4$ linadv noise

(e) $\epsilon = 0.1$ signflip noise (f) $\epsilon = 0.2$ signflip noise (g) $\epsilon = 0.3$ signflip noise (h) $\epsilon = 0.4$ signflip noise

(i) $\epsilon = 0.1$ uninoise noise (j) $\epsilon = 0.2$ uninoise noise (k) $\epsilon = 0.3$ uninoise noise (l) $\epsilon = 0.4$ uninoise noise

(m) $\epsilon = 0.1$ mixture noise (n) $\epsilon = 0.2$ mixture noise (o) $\epsilon = 0.3$ mixture noise (p) $\epsilon = 0.4$ mixture noise

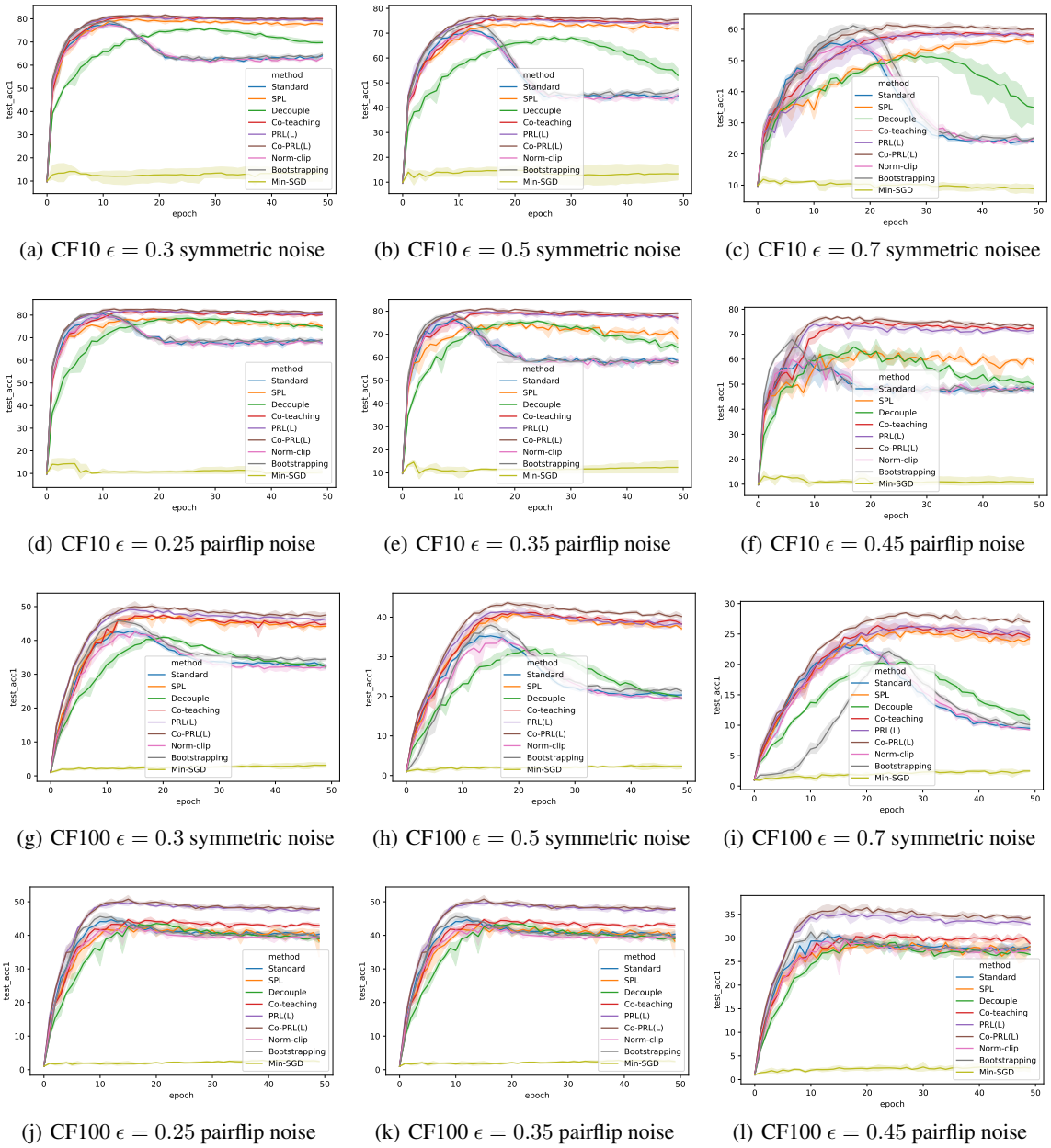
(q) $\epsilon = 0.1$ pairflip noise (r) $\epsilon = 0.2$ pairflip noise (s) $\epsilon = 0.3$ pairflip noise (t) $\epsilon = 0.4$ pairflip noise

Figure 2. Testing R-square for CelebA during the training phase.

7. Proofs

7.1. Proof of Convergence of Biased SGD

We gave the proof of the theorem of how biased gradient affect the final convergence of SGD. We introduce several assumptions and definition first:



(a) CF10 $\epsilon = 0.3$ symmetric noise (b) CF10 $\epsilon = 0.5$ symmetric noise (c) CF10 $\epsilon = 0.7$ symmetric noisee

(d) CF10 $\epsilon = 0.25$ pairflip noise (e) CF10 $\epsilon = 0.35$ pairflip noise (f) CF10 $\epsilon = 0.45$ pairflip noise

(g) CF100 $\epsilon = 0.3$ symmetric noise (h) CF100 $\epsilon = 0.5$ symmetric noise (i) CF100 $\epsilon = 0.7$ symmetric noise

(j) CF100 $\epsilon = 0.25$ pairflip noise (k) CF100 $\epsilon = 0.35$ pairflip noise (l) CF100 $\epsilon = 0.45$ pairflip noise

Figure 3. CIFAR10 and CIFAR100 Testing Curve During Training. X axis represents the epoch number, Y axis represents the testing accuracy. The shadow represents the confidence interval, which is calculated across 3 random seed. As we see, PRL(L), and Co-PRL(L) are robust against different types of corruptions.

Assumption 1 (L-smoothness) *The function ϕ: $\mathbb{R}^d \to \mathbb{R}$ is differentiable and there exists a constant $L > 0$ such that for all $\theta_1, \theta_2 \in \mathbb{R}^d$, we have $\phi(\theta_2) \leq \phi(\theta_1) + \langle \nabla\phi(\theta_1), \theta_2 - \theta_1 \rangle + \frac{L}{2}\|\theta_2 - \theta_1\|^2$*

Definition 1 (Biased gradient oracle) *A map $\mathbf{g} : \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^d$, such that $\mathbf{g}(\theta, \xi) = \nabla\phi(\theta) + \mathbf{b}(\theta, \xi) + \mathbf{n}(\theta, \xi)$ for a bias $\mathbf{b} : \mathbb{R}^d \to \mathbb{R}^d$ and zero-mean noise $\mathbf{n} : \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^d$, that is $\mathbb{E}_\xi \mathbf{n}(\theta, \xi) = 0$.*

Compared to standard stochastic gradient oracle, the above definition introduces the bias term $\mathbf{b}$. In noisy-label settings, the $\mathbf{b}$ is generated by the data with corrupted labels.

Assumption 2 (σ-Bounded noise) *There exists constants $\sigma > 0$, such that $\mathbb{E}_\xi \|\mathbf{n}(\theta, \xi)\|^2 \leq \sigma, \quad \forall \theta \in \mathbb{R}^d$*

Method	Standard	PRL(G)	PRL(L)
CF10-Pair-45%	37.03s	145.55s	54.80s

Table 3. Execution Time of Single Epoch in CIFAR-10 Data

Assumption 3 (ζ-Bounded bias) *There exists constants $\zeta > 0$, such that for any ξ, we have $\|\mathbf{b}(\theta, \xi)\|^2 \leq \zeta^2, \quad \forall \theta \in \mathbb{R}^d$*

For simplicity, assume the learning rate is constant γ, then in every iteration, the biased SGD performs update $\theta_{t+1} \leftarrow \theta_t - \gamma_t \mathbf{g}(\theta_t, \xi)$. Then the following theorem showed the gradient norm convergence with biased SGD.

Theorem 1 (Convergence of Biased SGD(formal)) *Under assumptions 1, 2, 3, define $F = \phi(\theta_0) - \phi^*$and step size $\gamma = \min\left\{\frac{1}{L}, (\sqrt{\frac{LF}{\sigma T}})\right\}$, denote the desired accuracy as k, then*

$$T = \mathcal{O}\left(\frac{1}{k} + \frac{\sigma^2}{k^2}\right)$$

iterations are sufficient to obtain $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\theta_t)\|^2\right) = \mathcal{O}(k + \zeta^2)$.

Remark 1 *Let $k = \zeta^2$, $T = \mathcal{O}\left(\frac{1}{\zeta^2} + \frac{\sigma^2}{\zeta^4}\right)$ iterations is sufficient to get $\min_{t\in[T]} \mathbb{E}\left(\|\nabla\phi(\theta_t)\|^2\right) = \mathcal{O}(\zeta^2)$, and performing more iterations does not improve the accuracy in terms of convergence.*

Since this is a standard results, more general results are showed in (Hu et al., 2020; Ajalloeian & Stich, 2020). For the sake of completeness, we provide the proof here.
Proof: by L-smooth, we have:

$$\phi(\theta_2) \leq \phi(\theta_1) + \langle \nabla\phi(\theta_1), \theta_2 - \theta_1\rangle + \frac{L}{2}\|\theta_2 - \theta_1\|^2$$

by using $\gamma \leq \frac{1}{L}$, we have

$$\begin{aligned}
\mathbb{E}\phi\left(\theta_{\mathbf{1}t+1}\right) &\leq \phi\left(\theta_{\mathbf{1}t}\right) - \gamma\left\langle\nabla\phi\left(\theta_{\mathbf{1}t}\right), \mathbb{E}\mathbf{g}_t\right\rangle + \frac{\gamma^2 L}{2}\left(\mathbb{E}\left\|\mathbf{g}_t - \mathbb{E}\mathbf{g}_t\right\|^2 + \mathbb{E}\left\|\mathbb{E}\mathbf{g}_t\right\|^2\right) \\
&= \phi\left(\theta_{\mathbf{1}t}\right) - \gamma\left\langle\nabla\phi\left(\theta_{\mathbf{1}t}\right), \nabla\phi\left(\theta_{\mathbf{1}t}\right) + \mathbf{b}_t\right\rangle + \frac{\gamma^2 L}{2}\left(\mathbb{E}\left\|\mathbf{n}_t\right\|^2 + \mathbb{E}\left\|\nabla\phi\left(\theta_{\mathbf{1}t}\right) + \mathbf{b}_t\right\|^2\right) \\
&\leq \phi\left(\theta_{\mathbf{1}t}\right) + \frac{\gamma}{2}\left(-2\left\langle\nabla\phi\left(\theta_{\mathbf{1}t}\right), \nabla\phi\left(\theta_{\mathbf{1}t}\right) + \mathbf{b}_t\right\rangle + \left\|\nabla\phi\left(\theta_{\mathbf{1}t}\right) + \mathbf{b}_t\right\|^2\right) + \frac{\gamma^2 L}{2}\mathbb{E}\left\|\mathbf{n}_t\right\|^2 \\
&= \phi\left(\theta_{\mathbf{1}t}\right) + \frac{\gamma}{2}\left(-\left\|\nabla\phi\left(\theta_{\mathbf{1}t}\right)\right\|^2 + \left\|\mathbf{b}_t\right\|^2\right) + \frac{\gamma^2 L}{2}\mathbb{E}\left\|\mathbf{n}_t\right\|^2
\end{aligned}$$

Since we have $\|\mathbf{b}_t\|^2 \leq \zeta^2$, $\|\mathbf{n}_t\|^2 \leq \sigma^2$, by plug in the learning rate constraint, we have

$$\begin{aligned}
\mathbb{E}\phi\left(\theta_{\mathbf{1}t+1}\right) &\leq \phi\left(\theta_{\mathbf{1}t}\right) - \frac{\gamma}{2}\left\|\nabla\phi\left(\theta_{\mathbf{1}t}\right)\right\|^2 + \frac{\gamma}{2}\zeta^2 + \frac{\gamma^2 L}{2}\sigma^2 \\
\mathbb{E}\phi\left(\theta_{\mathbf{1}t+1}\right) - \phi\left(\theta_{\mathbf{1}t}\right) &\leq -\frac{\gamma}{2}\left\|\nabla\phi\left(\theta_{\mathbf{1}t}\right)\right\|^2 + \frac{\gamma}{2}\zeta^2 + \frac{\gamma^2 L}{2}\sigma^2
\end{aligned}$$

Then, removing the gradient norm to left hand side, and sum it across different iterations, we could get

$$\frac{1}{2T}\sum_{t=0}^{T-1}\mathbb{E}\|\phi\left(\theta_{\mathbf{1}t}\right)\| \leq \frac{F}{T\gamma} + \frac{\zeta^2}{2} + \frac{\gamma L\sigma^2}{2}$$

Take the minimum respect to t and substitute the learning rate condition will directly get the results.

7.2. Proof of Corollary 1

We first prove the gradient estimation error.

Denote $\tilde{\mathbf{G}}$ to be the set of corrupted minibatch, $\mathbf{G}$ to be the set of original clean minibatch and we have $|\mathbf{G}| = |\tilde{\mathbf{G}}| = m$. Let $\mathbf{N}$ to be the set of remaining data and according to our algorithm, the remaining data has the size $|\mathbf{N}| = n = (1-\epsilon)m$. Define $\mathbf{A}$ to be the set of individual clean gradient, which is not discarded by algorithm 1. $\mathbf{B}$ to be the set of individual corrupted gradient, which is not discarded. According to our definition, we have $\mathbf{N} = \mathbf{A} \cup \mathbf{B}$. $\mathbf{AD}$ to be the set of individual good gradient, which is discarded, $\mathbf{AR}$ to be the set of individual good gradient, which is replaced by corrupted data. We have $\mathbf{G} = \mathbf{A} \cup \mathbf{AD} \cup \mathbf{AR}$. $\mathbf{BD}$ is the set of individual corrupted gradient, which is discarded by our algorithm. Denote the good gradient to be $\mathbf{g}_i = \alpha_i \mathbf{W}_i$, and the bad gradient to be $\tilde{\mathbf{g}}_i$, according to our assumption, we have $\|\tilde{\mathbf{g}}_i\| \leq L$.

Now, we have the l2 norm error:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &= \|\frac{1}{m}\sum_{i\in\mathbf{G}}^{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i=1}^{m}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \\
&\leq \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i\| + \|\frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \\
&\leq \|\sum_{\mathbf{A}}\frac{m-n}{nm}\mathbf{g}_i + \sum_{\mathbf{AD}}\frac{1}{m}\mathbf{g}_i + \sum_{\mathbf{AR}}\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\| \\
&\leq \sum_{\mathbf{A}}\|\frac{m-n}{nm}\mathbf{g}_i\| + \sum_{\mathbf{AD}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{AR}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\|
\end{aligned}$$

By using the filtering algorithm, we could guarantee that $\|\tilde{\mathbf{g}}_i\| \leq L$. Let $|\mathbf{A}| = x$, we have $|\mathbf{B}| = n - x = (1-\epsilon)m - x$, $|\mathbf{AR}| = m - n = \epsilon m$, $|\mathbf{AD}| = m - |\mathbf{A}| - |\mathbf{AR}| = m - x - (m-n) = n - x = (1-\epsilon)m - x$. Thus, we have:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq x\frac{m-n}{nm}L + (n-x)\frac{1}{m}L + (m-n)\frac{1}{m}L + (n-x)\frac{1}{n}L \\
&\leq x(\frac{m-n}{nm} - \frac{1}{m})L + n\frac{1}{m}L + (m-n)\frac{1}{m}L + (n-x)\frac{1}{n}L \\
&= \frac{1}{m}(\frac{2\epsilon-1}{1-\epsilon})xL + L + L - \frac{1}{n}xL \\
&= xL(\frac{2\epsilon-2}{n}) + 2L
\end{aligned}$$

To minimize the upper bound, we need x to be as small as possible since $2\epsilon - 2 < 1$. According to our problem setting, we have $x = n - m\epsilon \leq (1-2\epsilon)m$, substitute back we have:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq (1-2\epsilon)Lm(\frac{2\epsilon-2}{n}) + 2L \\
&= \frac{1-2\epsilon}{1-\epsilon}2L + 2L \\
&= 4L - \frac{\epsilon}{1-\epsilon}2L
\end{aligned}$$

Since $\epsilon < 0.5$, we use tylor expansion on $\frac{\epsilon}{1-\epsilon}$, by ignoring the high-order terms, we have

$$\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| = \mathcal{O}(\epsilon L)$$

Note, if the Lipschitz continuous assumption does not hold, then L should be dimension dependent (i.e. $\sqrt{d}$).

Combining above gradient estimation error upper bound and Theorem 1, we could get the results in Corollary 1.

7.3. Proof of Randomized Filtering Algorithm

Lemma 1 (Gradient Estimation Error for Randomized Filtering) *Given a corrupted matrix $\tilde{\mathbf{G}} \in \mathbb{R}^{m\times d}$ generated in Problem 2. Let $\mathbf{G} \in \mathbb{R}^{m\times d}$ be the original clean gradient matrix. Suppose we are arbitrary select $n = (1-\epsilon)m$ rows from $\tilde{\mathbf{G}}$ to get remaining set $\mathbf{N} \in \mathbb{R}^{n\times d}$. Let μ to be the empirical mean function, assume the clean gradient before loss layer has bounded operator norm: $\|\mathbf{W}\|_{op} \leq C$, the maximum clean gradient in loss layer $\max_i \|\alpha_i\| = k$, the maximum corrupted gradient in loss layer $\max_i \|\delta_i\| = v$, assume $\epsilon < 0.5$, then we have:*

$$\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| \leq Ck\frac{3\epsilon - 4\epsilon^2}{1-\epsilon} + Cv\frac{\epsilon}{1-\epsilon}$$

7.4. Proof of lemma 1

Denote $\tilde{\mathbf{G}}$ to be the set of corrupted minibatch, $\mathbf{G}$ to be the set of original clean minibatch and we have $|\mathbf{G}| = |\tilde{\mathbf{G}}| = m$. Let $\mathbf{N}$ to be the set of remaining data and according to our algorithm, the remaining data has the size $|\mathbf{N}| = n = (1-\epsilon)m$. Define $\mathbf{A}$ to be the set of individual clean gradient, which is not discarded by any filtering algorithm. $\mathbf{B}$ to be the set of individual corrupted gradient, which is not discarded. According to our definition, we have $\mathbf{N} = \mathbf{A} \cup \mathbf{B}$. $\mathbf{AD}$ to be the set of individual good gradient, which is discarded, $\mathbf{AR}$ to be the set of individual good gradient, which is replaced by corrupted data. We have $\mathbf{G} = \mathbf{A} \cup \mathbf{AD} \cup \mathbf{AR}$. $\mathbf{BD}$ is the set of individual corrupted gradient, which is discarded by our algorithm. Denote the good gradient to be $\mathbf{g}_i = \alpha_i \mathbf{W}_i$, and the bad gradient to be $\tilde{\mathbf{g}}_i = \delta_i \mathbf{W}_i$, according to our assumption, we have $\|\mathbf{W}_i\|_{op} \leq C$.

Now, we have the l2 norm error:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &= \|\frac{1}{m}\sum_{i\in\mathbf{G}}^{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i=1}^{m}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \left(\frac{1}{n}\sum_{i\in\mathbf{A}}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\right)\| \\
&= \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i - \frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \\
&\leq \|\frac{1}{n}\sum_{i\in\mathbf{A}}(\frac{n-m}{m})\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AD}}\frac{n}{m}\mathbf{g}_i + \frac{1}{n}\sum_{i\in\mathbf{AR}}\frac{n}{m}\mathbf{g}_i\| + \|\frac{1}{n}\sum_{i\in\mathbf{B}}\tilde{\mathbf{g}}_i\| \qquad (1)
\end{aligned}$$

Let $|\mathbf{A}| = x$, we have $|\mathbf{B}| = n - x = (1-\epsilon)m - x$, $|\mathbf{AR}| = m - n = \epsilon m$, $|\mathbf{AD}| = m - |\mathbf{A}| - |\mathbf{AR}| = m - x - (m-n) = n - x = (1-\epsilon)m - x$. Thus, we have:

$$\begin{aligned}
\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq \|\sum_{\mathbf{A}}\frac{m-n}{nm}\mathbf{g}_i + \sum_{\mathbf{AD}}\frac{1}{m}\mathbf{g}_i + \sum_{\mathbf{AR}}\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\| \\
&\leq \sum_{\mathbf{A}}\|\frac{m-n}{nm}\mathbf{g}_i\| + \sum_{\mathbf{AD}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{AR}}\|\frac{1}{m}\mathbf{g}_i\| + \sum_{\mathbf{B}}\frac{1}{n}\|\tilde{\mathbf{g}}_i\|
\end{aligned}$$

For individual gradient, according to the label corruption gradient definition in problem 2, assuming the $\|\mathbf{W}\|_{op} \leq C$, we have $\|\mathbf{g}_i\| \leq \|\alpha_i\|\|\mathbf{W}_i\|_{op} \leq C\|\alpha_i\|$. Also, denote $\max_i \|\alpha_i\| = k$, $\max_i \|\delta_i\| = v$, we have $\|\mathbf{g}_i\| \leq Ck$, $\|\tilde{\mathbf{g}}_i\| \leq Cv$.

$$\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| \leq Cx\frac{m-n}{nm}k + C(n-x)\frac{1}{m}k + C(m-n)\frac{1}{m}k + C(n-x)\frac{1}{n}v$$

Note the above upper bound holds for any x, thus, we would like to get the minimum of the upper bound respect to x. Rearrange the term, we have

$$\begin{aligned}\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq Cx(\frac{m-n}{nm} - \frac{1}{m})k + Cn\frac{1}{m}k + C(m-n)\frac{1}{m}k + C(n-x)\frac{1}{n}v \\ &= C\frac{1}{m}(\frac{2\epsilon-1}{1-\epsilon})xk + Ck + Cv - \frac{1}{n}Cxv \\ &= Cx\left(\frac{k(2\epsilon-1)}{m(1-\epsilon)} - \frac{v}{n}\right) + Ck + Cv \\ &= Cx\left(\frac{k(2\epsilon-1)-v}{m(1-\epsilon)}\right) + Ck + Cv\end{aligned}$$

Since when $\epsilon < 0.5$, $\frac{k(2\epsilon-1)-v}{m(1-\epsilon)} < 0$, we knew that x should be as small as possible to continue the bound. According to our algorithm, we knew $n - m\epsilon = m(1-\epsilon) - m\epsilon = (1-2\epsilon)m \leq x \leq n = (1-\epsilon)m$. Then, substitute $x = (1-2\epsilon)m$, we have

$$\begin{aligned}\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq Ck(1-2\epsilon)\frac{2\epsilon-1}{1-\epsilon} + Ck + Cv - Cv\frac{1-2\epsilon}{1-\epsilon} \\ &= Ck\frac{3\epsilon-4\epsilon^2}{1-\epsilon} + Cv\frac{\epsilon}{1-\epsilon}\end{aligned}$$

7.5. Proof of Theorem 2

According to algorithm2, we could guarantee that $v \leq k$. By lemma 1, we will have:

$$\begin{aligned}\|\mu(\mathbf{G}) - \mu(\mathbf{N})\| &\leq Ck\frac{3\epsilon-4\epsilon^2}{1-\epsilon} + Cv\frac{\epsilon}{1-\epsilon} \\ &\leq Ck\frac{4\epsilon-4\epsilon^2}{1-\epsilon} \\ &= 4\epsilon Ck \\ &\approx \mathcal{O}(\epsilon\sqrt{q})\text{(C is constant, k is the norm of } q\text{-dimensional vector)}\end{aligned}$$

7.6. Proof of Lemma 2

Assume we have a d class label $\mathbf{y} \in \mathcal{R}^d$, where $y_k = 1, y_i = 0, i \neq k$. We have two prediction $\mathbf{p} \in \mathcal{R}^d$, $\mathbf{q} \in \mathcal{R}^d$.

Assume we have a d class label $\mathbf{y} \in \mathbb{R}^d$, where $y_k = 1, y_i = 0, i \neq k$. With little abuse of notation, suppose we have two prediction $\mathbf{p} \in \mathbb{R}^d$, $\mathbf{q} \in \mathbb{R}^d$. Without loss of generality, we could assume that $\mathbf{p}_1$ has smaller cross entropy loss, which indicates $\mathbf{p}_k \geq \mathbf{q}_k$

For MSE, assume we have opposite result

$$\begin{aligned}&\|\mathbf{p} - \mathbf{y}\|^2 \geq \|\mathbf{q} - \mathbf{y}\|^2 \\ \Rightarrow &\sum_{i\neq k} p_i^2 + (1-p_k)^2 \geq \sum_{i\neq k} q_i^2 + (1-q_k)^2\end{aligned} \tag{2}$$

For each $p_i, i \neq k$, We have

$$Var(p_i) = E(p_i^2) - E(p_i)^2 = \frac{1}{d-1}\sum_{i\neq k} p_i^2 - \frac{1}{(d-1)^2}(1-p_k)^2 \tag{3}$$

Then

$$\begin{aligned}
&\sum_{i\neq k} p_i^2 + (1-p_k)^2 \geq \sum_{i\neq k} q_i^2 + (1-q_k)^2 \\
\Rightarrow &Var_{i\neq k}(\mathbf{p}_i) + \frac{d}{(d-1)^2}(1-p_k)^2 \geq Var_{i\neq k}(\mathbf{q}_i) + \frac{d}{(d-1)^2}(1-q_k)^2 \\
\Rightarrow &Var_{i\neq k}(\mathbf{p}_i) - Var_{i\neq k}(\mathbf{q}_i) \geq \frac{d}{(d-1)^2}\left((1-q_k)^2 - (1-p_k)^2\right) \\
\Rightarrow &Var_{i\neq k}(\mathbf{p}_i) - Var_{i\neq k}(\mathbf{q}_i) \geq \frac{d}{(d-1)^2}\left((p_k-q_k)(2-p_k-q_k)\right)
\end{aligned} \tag{4}$$

References

Ajalloeian, A. and Stich, S. U. Analysis of sgd with biased gradient estimators. *arXiv preprint arXiv:2008.00051*, 2020.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pp. 8527–8537, 2018.

Hu, Y., Zhang, S., Chen, X., and He, N. Biased stochastic gradient descent for conditional stochastic optimization. *arXiv preprint arXiv:2002.10790*, 2020.