Learning with Bad Training Data via Iterative Trimmed Loss Minimization

Yanyao Shen [1] **Sujay Sanghavi** [1]

Abstract

In this paper, we study a simple and generic framework to tackle the problem of learning model parameters when a fraction of the training samples are corrupted. We first make a simple observation: in a variety of such settings, the evolution of training accuracy (as a function of training epochs) is different for clean and bad samples. Based on this we propose to iteratively minimize the *trimmed loss*, by alternating between (a) selecting samples with lowest current loss, and (b) retraining a model on only these samples. We prove that this process recovers the ground truth (with linear convergence rate) in generalized linear models with standard statistical assumptions. Experimentally, we demonstrate its effectiveness in three settings: (a) deep image classifiers with errors only in labels, (b) generative adversarial networks with bad training images, and (c) deep image classifiers with adversarial (image, label) pairs (i.e., backdoor attacks). For the well-studied setting of random label noise, our algorithm achieves state-of-the-art performance without having access to any a-priori guaranteed clean samples.

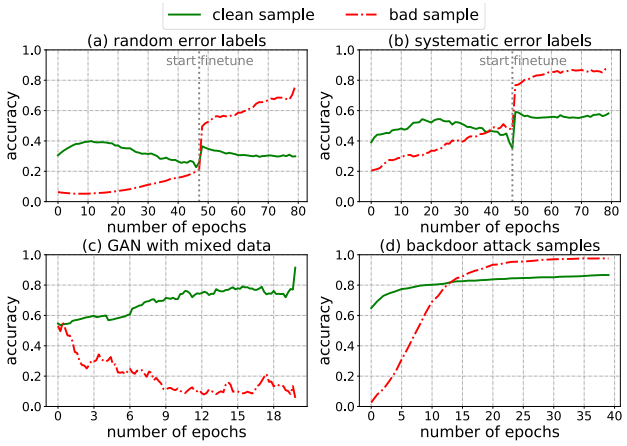


Figure 1. **Observation:** Evolution of model accuracy for clean and bad samples, as a function of training epochs, for four different tainted data settings: (a) classification for CIFAR-10 with 40% random errors in labels, (b) classification for CIFAR-10 with 40% systematic errors in labels, (c) DC-GAN trained on unlabeled mixture of 70% MNIST images with 30% Fashion-MNIST images, (d) backdoor attack on classification for CIFAR-10 with 250 watermarked backdoor attach samples as described in (Tran et al., 2018). The CIFAR-10 classifications are done using the WideResNet-16 of (Zagoruyko & Komodakis, 2016). In all instances models are trained on the respective tainted data. **Early on, models are more accurate on the good samples.**

1. Introduction

State of the art accuracy in several machine learning problems now requires training very large models (i.e. with lots of parameters) using very large training data sets. Such an approach can be very sensitive to the quality of the training data used; this is especially so when the models themselves are expressive enough to fit all data (good and bad) in way that may generalize poorly if data is bad. We are interested both in **poorly curated** datasets – label errors in supervised settings, and irrelevant samples in unsupervised settings – as well as situations like **backdoor attacks** (Gu et al., 2017) where a small number of adversarially altered samples (i.e. labels and features changed) can compromise security. These are well-recognized issues, and indeed several approaches exist for each kind of training data error; we summarize these in the related work section. However, these approaches are quite different, and in practice selecting which one to apply would need us to know / suspect the form of training data errors a-priori.

In this paper we provide a single, simple approach that can deal with several such tainted training data settings, based on a **key observation**. We consider the (common) setup where training proceeds in epochs / stages, and inspect the *evolution* of the accuracy of the model on the training samples – i.e. after each epoch, we take the model at that stage and see whether or not it makes an error for each of the training samples. Across several different settings with errors/corruptions in training data, we find that *the accuracy on "clean" samples is higher than on the "bad"*

[1]ECE Department, University of Texas at Austin, TX, USA. Correspondence to: Yanyao Shen <shenyanyao@utexas.edu>, Sujay Sanghavi <sanghavi@mail.utexas.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

samples, especially in the initial epochs of training. Figure 1 shows four different settings where this is the case. This observation suggests a natural approach: iteratively alternate between *(a)* filtering out samples with large (early) losses, and *(b)* re-training the model on the remaining samples. Both steps can be done in pretty much any machine learning setting: all that is needed is for one to be able to evaluate losses on training samples, and re-train models from a new set of samples.

Our approach is related to a classical statistical objective: minimizing the **trimmed loss**. Specifically, given a set of n samples, standard estimation / model-fitting involves choosing model parameters θ to minimize a loss function over all n samples; in contrast, the trimmed loss estimator involves jointly choosing a subset of αn samples and θ such that the loss on the subset is minimum (over all choices of subset and parameters). This objective is intractable in general; our approach can be viewed as an iterative way to minimize trimmed loss. We describe trimmed loss[1], its known properties, and some new results, in Section 3. Our approach (in Section 5) is thus an iterative way to find a trimmed loss estimator, hence we choose to call it Iterative Trimmed Loss Minimization (ITLM).

We propose ITLM as a generic approach to the problem of training with tainted data and investigate its performance both theoretically and empirically. More specifically, **our contributions** include:
(a) In **Section 5**, we analyze ITLM applied to a setting where the clean samples come from a ground-truth generalized linear model, and the bad samples have the response variables being (i) arbitrary corruption; (ii) random output; (iii) mixture output. We show ITLM converges at least linearly to the ground truth model in all these settings. Our theoretic findings are further verified with synthetic experiments in **Section 6.1**.We also include a basic asymptotic property for general functions in **Section 3**;
(b) In **Section 6.2**, we show ITLM can be applied to classification problems with bad labels. For CIFAR-10 classification with random labels, ITLM performs better than previous state-of-the-art results, without using any identified clean sample;
(c) In **Section 6.3 and 6.4**, we successfully apply ITLM to image generation task with bad images and classification task with adversarial (image, label) pairs (backdoor attacks).

Notations For integer m, $[m]$ denotes the set $\{0, \cdots, m-1\}$. For real number a, $\lfloor a \rfloor$ denotes the maximum integer no greater than a. $\sigma_{\min}$ and $\sigma_{\max}$ are the minimum/maximum eigenvalues. $a \wedge b$ and $a \vee b$ are shorthands for $\min\{a, b\}, \max\{a, b\}$. $|S|$ is the cardinality for set S. For two sets S_1, S_2, $S_1 \backslash S_2$ is the set of elements in S_1 but not in S_2. The term *w.h.p.* means *with probability at least* $1 - n^{-c}$ *where* c *is an aribatrary constant*.

[1] Our framework sounds initially similar to EM-style algorithms like k-means. Note however that EM needs to postulate a model for all the data points, while we search over a subset and do not worry about the loss on corrupted points. We are alternating between a simple search over subsets and a fitting problem on only the *selected* subset; this is not an instance of EM.

2. Related Work

There is a vast literature on bad training data problems. We classify the most related work from classic statistics to machine learning frontiers into the following four genres.

Robust regression There are several classes of robust estimators (Huber, 2011). Among them, Least Trimmed Square (LTS) estimator (Rousseeuw, 1984) has high breakdown point and is sample efficient. Following the idea of LTS, several recent works provide algorithmic solutions and analyze their theoretical guarantees (Bhatia et al., 2015; Vainsencher et al., 2017; Yang et al., 2018). Different from previous works, we provide a fine characterization of the convergence in several settings, which connects to the problems of noisy labels/adversarial backdoor attack in practice. We also experimentally explore the overall approach for more complex tasks with deep neural network models. Notice that our approach is certainly not the only algorithm solution to finding least trimmed estimators. For example, see (Hössjer, 1995; Rousseeuw & Van Driessen, 2006; Shen et al., 2013) for algorithm solutions finding the least trimmed loss estimator in linear regression setting. However, compared to other works, our approach is more scalable, and not sensitive to the selection of loss functions. Another line of recent work on robust regression consider strong robustness where the adversary poisons both the inputs and outputs, in both low-dimensional (Diakonikolas et al., 2018; Prasad et al., 2018; Klivans et al., 2018) and high dimensional (Chen et al., 2013; Balakrishnan et al., 2017a; Liu et al., 2018b) settings. These algorithms usually require much more computation compared with, e.g., the algorithm we consider in this paper.

Mixed linear regression Alternating minimization type algorithms are used for mixed linear regression with convergence guarantee (Yi et al., 2014; Balakrishnan et al., 2017b), in the setting of two mixtures. For multiple mixture setting, techniques including tensor decomposition are used (Yi et al., 2016; Zhong et al., 2016; Sedghi et al., 2016; Li & Liang, 2018), but require either high sample complexity or high computation complexity (especially when number of mixtures is large). On the other hand, (Ray et al., 2018) studies finding a single component in mixture problems using a particular type of side information.

Noisy label problems Classification tasks with noisy labels are also of wide interest. (Frénay et al., 2014) gives an overview of the related methods. Theoretical guarantee for noisy binary classification has been studied under different settings (Scott et al., 2013; Natarajan et al., 2013; Menon et al., 2016). More recently, noisy label problem has been studied for DNNs. (Reed et al., 2014) and (Malach & Shalev-Shwartz, 2017) develop the idea of bootstrapping and query-by-committee for DNNs. On the other hand, (Khetan et al., 2018) and (Zhang & Sabuncu, 2018) provide new losses for training under the noise. (Sukhbaatar & Fergus, 2014) adds a noise layer into the training process, while (Ren et al., 2018) provides a meta-algorithm for learning the weights of all samples by heavily referencing to a clean validation data during training. (Jiang et al., 2017) proposes a data-driven curriculum learning approach.

Defending backdoor attack Several recent works defend against backdoor attack samples (Gu et al., 2017). (Tran et al., 2018) proposes using spectral signature, where they calculate the top singular vector of a certain layer's representation for all the samples. (Liu et al., 2018a) proposes pruning DNNs based on the belief that backdoor samples exploit spare capacity. (Wang et al.) uses a reverse engineering approach. (Chen et al., 2018) detect by activation clustering. While the adversary is not allowed to train the model, these approaches do not exploit the evolution of the training accuracy for detecting backdoor samples.

3. Setup and (Exact) Trimmed Loss Estimator

We now describe the least trimmed loss estimator in general. Let $s_1, \ldots, s_n$ be the samples, θ the model parameters to be learnt, and loss function $f_\theta(\cdot)$. With this setting, the standard approach is to minimize the total loss of all samples, i.e. $\min_\theta \sum_i f_\theta(s_i)$. In contrast, the least trimmed loss estimator is given by

$$\hat{\theta}^{(\mathrm{TL})} = \arg\min_{\theta\in\mathcal{B}} \min_{S:|S|=\lfloor \alpha n\rfloor} \sum_{i\in S} f_\theta(s_i),$$

For finding $\hat{\theta}^{(\mathrm{TL})}$ we need to minimize over *both* the set S of size $\lfloor \alpha n\rfloor$ – where $\alpha \in (0,1)$ is the fraction of samples we want to fit – and the set of parameters θ. In general solving for the least trimmed loss estimator is hard, even in the linear regression setting (Mount et al., 2014), i.e., even when $s = (x, y)$ and $f_\theta(x, y) = (y - \theta^\top x)^2$. Nevertheless, its statistical efficiency has been studied. In the linear setting, it has a breakdown point of $1/2$ asymptotically (Huber, 2011), and is consistent (Víšek, 2006), i.e., $\hat{\theta}^{(\mathrm{TL})} \to \theta^\star$ in probability as $n \to \infty$. (Čížek, 2008) also shows this property for more general function classes.

We now present a basic result for more general non-linear functions. Let $\mathcal{B}$ be a compact parametric space, and all samples are i.i.d. generated following certain distribution. For $\theta \in \mathcal{B}$, let D_θ, d_θ be the distribution and density function of $f_\theta(s)$. Let $S(\theta) = \mathbb{E}_s[f_\theta(s)]$ be the population loss, and let $S_n(\theta) = \frac{1}{n}\sum_{i=1}^n f_\theta(s_i)$ be the empirical loss. Define $F(\theta) := \mathbb{E}\left[f_\theta(s)\mathbf{I}(f_\theta(s) \le D_\theta^{-1}(\alpha))\right]$ as the population trimmed loss. Let $\mathcal{U}(\theta,\epsilon) := \{\tilde{\theta} \mid |S(\tilde{\theta}) - S(\theta)| < \epsilon, \tilde{\theta} \in \mathcal{B}\}$ be the set of parameters with population loss close to θ. We require the following two natural assumptions:

Assumption 1 (Identification condition for $\theta^\star$)**.** *For every $\epsilon > 0$ there exists a $\delta > 0$ such that if $\theta \in \mathcal{B}\backslash\mathcal{U}(\theta^\star,\epsilon)$, we have that $F(\theta) - F(\theta^\star) > \delta$.*

Assumption 2 (Regularity conditions)**.** *D_θ is absolutely continuous for any $\theta \in \mathcal{B}$. d_θ is bounded uniformly in $\theta \in \mathcal{B}$, and is locally positive in a neighborhood of its α-quantile. $f_\theta(s)$ is differentiable in θ for $\theta \in \mathcal{U}(\theta^\star,\epsilon)$, for some $\epsilon > 0$.*

The identification condition identifies $\theta^\star$ as achieving the global minimum on the population trimmed loss. The regularity conditions are standard and very general. Based on these two assumptions, we show that TL is consistent with $\theta^\star$ in empirical loss.

Lemma 3. *Under Assumptions 1 and 2, the estimator $\hat{\theta}^{(\mathrm{TL})}$ satisfies: $\left|S_n(\hat{\theta}^{(\mathrm{TL})}) - S_n(\theta^\star)\right| \to 0$ with probability 1, as $n \to \infty$.*

4. Iterative Trimmed Loss Minimization

Our approach to (attempt to) minimize the trimmed loss, by alternating between minimizing over S and θ is described below.

Algorithm 1 Iterative Trimmed Loss Minimization (`ITLM`)

input: samples $\{s_i\}_{i=1}^n$, number of rounds T, fraction of samples α
(optional) initialize: $\theta_0 \leftarrow \arg\min_\theta \sum_{i\in[n]} f_\theta(s_i)$
for $t = 0, \cdots, T-1$ **do**
  choose samples with smallest current loss f_{θ_t}

$$S_t \leftarrow \arg\min_{S:|S|=\lfloor \alpha n\rfloor} \sum_{i\in S} f_{\theta_t}(s_i)$$

  $\theta_{t+1} = \texttt{ModelUpdate}(\theta_t, S_t, t)$
return: θ_T

Here $\texttt{ModelUpdate}(\theta_t, S_t, t)$ refers to the process of finding a new θ given sample set S_t, using θ_t as the initial value (if needed) in the update algorithm, and also the round number t. For example this could just be the (original, naive) estimator that minimizes the loss over all the samples given to it, which is now S_t.

In this paper we will use batch stochastic gradient as our

model update procedure, so we now describe this for completeness.

Algorithm 2 `BatchSGD_ModelUpdate`(θ, S, t)

input: initial parameter θ, set S, round t
choose: step size η, number M of gradient steps, batch size N
(optional) re-initialize θ^0 randomly
for $j = 1, \cdots, M$ **do**
  $B_j \leftarrow$ `random_subset`(S, N)
  $\theta^j \leftarrow \theta^{j-1} - \eta \left(\frac{1}{N} \sum_{i \in B_j} \nabla_\theta f_{\theta^{j-1}}(s_i) \right)$
return: θ^M

Note that for different settings, we use the same procedure as described in Algorithm 1 and 2, but may select different hyper-parameters. We will clarify the alternatives we use in each part.

5. Theoretical Guarantees for Generalized Linear Models

We now analyze ITLM for generalized linear models with errors in the outputs (but not in the features): we are given samples each of the form (x, y) such that

$$\begin{aligned} y &= \omega(\phi(x)^\top \cdot \theta^\star) + e, && \text{(clean samples)} \\ y &= r + e, && \text{(bad samples)} \end{aligned} \tag{1}$$

Here x represents the inputs, y the output, embedding function ϕ and link function w are known (and possibly non-linear) [2], e is random subgaussian noise with parameter σ^2 (Vershynin, 2010), and $\theta^\star$ is the ground truth. Thus there are errors in outputs y of bad samples, but not the features. Let $\alpha^\star$ be the fraction of clean samples in the dataset.

For ITLM , we use squared loss, i.e. $f_\theta(x, y) = (y - \omega(\phi(x)^\top \cdot \theta))^2$. We will also assume the feature matrices are regular, which is defined below.

Definition 4. *Let $\Phi(X) \in \mathbb{R}^{n \times d}$ be the feature matrix for all samples, where the ith row is $\phi(x_i)^\top$. Let $\mathcal{W}_k = \{W \in \mathbb{R}^{n \times n} | W_{i,j} = 0, W_{i,i} \in \{0, 1\}, \mathtt{Tr}(W) = k\}$. Define*

$$\psi^-(k) = \min_{W: W \in \mathcal{W}_k} \sigma_{\min} \left(\Phi(X)^\top W \Phi(X) \right),$$
$$\psi^+(k) = \max_{W: W \in \mathcal{W}_k} \sigma_{\max} \left(\Phi(X)^\top W \Phi(X) \right).$$

We say that $\Phi(X)$ is a regular feature matrix if for $k = \alpha n$, $\alpha \in [c, 1]$, $\psi^-(k) = \psi^+(k) = \Theta(n)$ for $n = \Omega(d \log d)$.

Regularity states that every large enough subset of samples results in a $\Phi(X)$ that is well conditioned. This holds under several natural settings, please see Appendix for more discussion. We now first present a one-step update lemma for the linear case.

Lemma 5 (linear case). *Assume $\omega(x) = x$ and we are using ITLM with α. The (for large enough M and small η in Algorithm 2), the following holds per round update w.h.p.:*

$$\|\theta_{t+1} - \theta^\star\|_2 \le \frac{\sqrt{2}\gamma_t}{\psi^-(\alpha n)} \|\theta_t - \theta^\star\|_2 + \frac{\sqrt{2}\varphi_t + c\xi_t\sigma}{\psi^-(\alpha n)},$$

where $\varphi_t = \left\| \sum_{i \in S_t \setminus S^\star} (\phi(x_i)^\top \theta_t - r_i - e_i) \phi(x_i) \right\|_2$, *and* $\gamma_t = \psi^+(|S_t \setminus S^\star|)$, $\xi_t = \sqrt{\sum_{i=1}^n \|\phi(x_i)\|_2^2 \log n}$.

This Lemma 5 bounds the error in the next step based on the error in the current step, how mismatched the set S_t is as compared to the true good set $S^\star$, and the regularity parameters. The following does the same for the more general non-linear case.

Lemma 6 (non-linear case). *Assume $\omega : \mathbb{R} \to \mathbb{R}$ monotone and differentiable. Assume $\omega'(u) \in [a, b]$ for all $u \in \mathbb{R}$, where a, b are positive constants. Then, for ITLM with α (and $M = 1$ and $N = |S|$ in Algorithm 2), w.h.p.,*

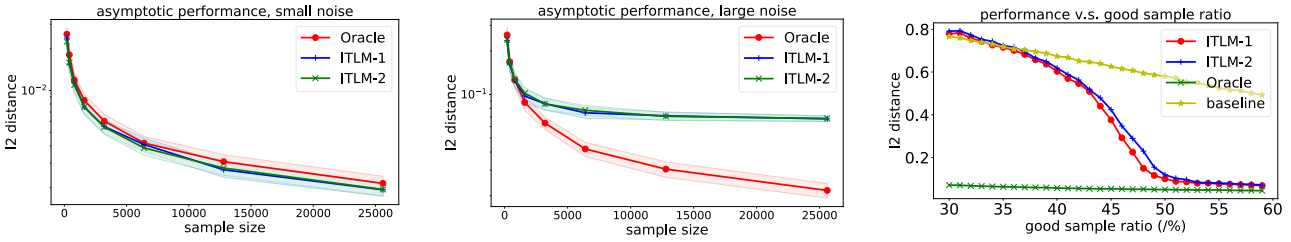
$$\|\theta_{t+1} - \theta^\star\|_2 \le \left(1 - \frac{\eta}{\alpha n} a^2 \psi^-(\alpha n)\right) \|\theta_t - \theta^\star\|_2 + \eta \frac{\tilde{\varphi}_t + \xi_t b \sigma}{\alpha n}$$

where ξ_t is the same as in Lemma 5, and $\tilde{\varphi}_t = \left\| \sum_{i \in S_t \setminus S^\star} \left(w(\phi(x_i)^\top \theta^\star) - r_i - e_i \right) w'(\phi(x_i)^\top \theta^\star) \phi(x_i) \right\|$.

Remarks: A few comments
(1) Lemma 5 and Lemma 6 directly lead to the consistency of the algorithm in the clean data setting ($\alpha^\star = 1$). This is because $|S_t \setminus S^\star| = 0$, which makes both γ_t and $\varphi_t(\tilde{\varphi}_t)$ become zero. Moreover, ξ_t is sublinear in n, and can be treated as the statistical error. While this consistency property is not very surprising (remind that Section 3 shows TL estimator has consistent performance for very general class of functions), the update property helps us better analyze convergence behavior in multiple corruption settings.
(2) In (Bhatia et al., 2015), convergence of the parameter is characterized by the linear convergence of $\sum_{i \in S_t} r_i^2$ with constant rate. Here, by directly characterizing the convergence in parameter space, we gain several additional benefits: (a) generality: we can directly analyze our result under several settings, including arbitrary corruption, random output, and mixture output; (b) finer chracterization: we see that the rate depends on $\alpha_t / \psi^-(\alpha n)$, which goes down as long as $|S_t \setminus S^\star|$ goes down. We can have a finer characterization of the convergence, e.g., super-linear convergence for the latter two corruption settings.

Next, we specialize our analysis into several bad training data settings. In the main paper we state Theorem 7 and Theorem 8 for the linear setting $\omega(x) = x$, while Theorems

[2] In neural network models, for example, $\phi(x)$ would represent the output of the final representation layer, and we assume that the parameters of the previous layers are fixed.



(a) (b) (c)

Figure 2. Synthetic experiments: **(a):** $\|\theta^\star - \theta_T\|_2$ v.s. sample size under small measurement noise; **(b):** under large measurement noise; **(c):** $\|\theta^\star - \theta_T\|_2$ v.s. different good sample ratio. ITLM -1: ITLM with large M (full update per round); ITLM -2: ITLM with $M = 1$.

9 and 10 in the appendix represent the same for the non-linear setting.

Our first result shows per-round convergence when a fraction $1 - \alpha^\star$ samples are either arbitrarily or randomly corrupted, and we choose the α in ITLM to be $\alpha \le \alpha^*$.

Theorem 7 (arbitrary/random corruptions). *Assume $\omega(x) = x$. We are given clean sample ratio $\alpha^\star > c_{\mathtt{th}}$, and ITLM with α such that $\alpha < \alpha^\star$ and sample size $n = \Omega(d \log d)$. Then w.h.p., we have:*

$$\|\theta^\star - \theta_{t+1}\|_2 \le \kappa_t \|\theta^\star - \theta_t\|_2 + c_1\sqrt{\kappa_t}\sigma + \frac{c_2\xi_t}{n}\sigma,$$

where $\kappa_t \le \frac{1}{2}$ when r is arbitrary, and $\kappa_t \le c\{\sqrt{\|\theta_t - \theta^\star\|_2^2 + \sigma^2} \vee \frac{\log n}{n}\}$ when r is random sub-Gaussian output. All the c constants depend on the regularity conditions.

Remark In both settings, we show at least linear convergence performance for per-round update. On the other hand, even in the infinite sample setting, the second term would not go to zero, which implies that our theoretic guarantee does not ensure consistency. In fact, we show in the next section that our analysis is tight, i.e., ITLM indeed gives inconsistent estimation. However, if the noise is small, ITLM will converge very closely to the ground truth. The proof is in Appendix.

We now consider the case when the data comes from a mixture model. We provide local convergence result that characterizes the performance of ITLM for this setting. More specifically, the full set of samples $S = [n]$ is split into m sets: $S = \cup_{j\in[m]} S_{(j)}$, each corresponding to samples from one mixture component, $|S_{(j)}| = \alpha^\star_{(j)} n$. The response variable y_i is given by:

$$y_i = \omega(\phi(x_i)^\top \theta^\star_{(j)}) + e_i, \text{ for } i \in S_{(j)}. \tag{2}$$

Fitted into the original framework, $r_i = \omega(\phi(x_i)^\top \theta^\star_{(j)})$ for some $j \in [m]\backslash\{0\}$. Similar to previous literatures (Yi et al., 2014; Zhong et al., 2016), we consider $\phi(x) \sim \mathcal{N}(0, I_d)$. We have the following local convergence guarantee in the mixture model setting:

Theorem 8 (mixed regression). *Assume $\omega(x) = x$ and consider ITLM with α. For the mixed regression setting in (2), suppose that for some component $j \in [m]$, we have that $\alpha < \alpha^\star_{(j)}$. Then, for $n = \Omega(d \log d)$, w.h.p., the next iterate θ_{t+1} of the algorithm satisfies*

$$\|\theta_{t+1} - \theta^\star_{(j)}\|_2 \le \kappa_t \|\theta_t - \theta^\star_{(j)}\|_2 + c_1\sqrt{\kappa_t}\sigma + \frac{c_2\xi_t}{n}\sigma,$$

where $\kappa_t \le c\left\{\frac{\sqrt{\|\theta_t - \theta^\star_{(j)}\|_2^2 + \sigma^2}}{\min_{k\in[m]\backslash\{j\}} \sqrt{\|\theta_t - \theta^\star_{(k)}\|_2^2 + \sigma^2}} \vee \frac{\log n}{n}\right\}$.

Remark Theorem 8 has a nearly linear dependence (sample complexity) on dimension d. In order to let $\kappa_0 < 1$, the iterate θ_0 in Algorithm 1 needs to satisfy $\|\theta_0 - \theta^\star_{(j)}\|_2 \le C(\alpha) \min_{k\in[m]\backslash\{j\}} \|\theta_0 - \theta^\star_{(k)}\|_2 - \sqrt{1 - C(\alpha)^2}\sigma$, where $C(\alpha) = \min\{\frac{c_3\alpha}{1-\alpha}, 1\}$. For α large enough such that $C(\alpha) = 1$, the condition on θ_0 does not depend on σ. However, for smaller α, the condition of θ_0 tolerates smaller noise. This is because, even if θ_0 is very close to $\theta^{(j)}$, when the noise and the density of samples from other mixture components are both high, the number of samples from other components selected by the current θ_0 would still be quite large, and the update will not converge to $\theta^{(j)}$.

6. Experiments

We first run simulations to verify and illustrate the theoretical guarantees we have in Section 5. Then, we present the result of our algorithm in several bad training data settings using DNNs, and compare them with the state-of-the-art results. Although deep networks have the capacity to fit bad samples as well as the clean samples, as we motivated in Section 1, the learning curve for clean samples is better than that of the bad samples (at least in early stages of training), which aligns with the linear setting. Besides, for the noisy label problem, we show that ITLM performs better than previous state-of-the-art methods where additional DNNs and additional clean samples are required. Our algorithm is simple to implement and requires neither of them. Training details are explained in Appendix.

Table 1. **Neural networks classification accuracy with random/systematic label error:** Performance for subsampled-MNIST, CIFAR-10 datasets as the ratio of clean samples varies. **Baseline** : Naive training using all the samples; **ITLM** : Our iterative update algorithm with $\alpha = \alpha^\star - 5\%$; **Oracle** : Training with all clean samples. **Centroid**: Filter out samples far away from the centroid for each label class; **1-step**: The first iteration of **ITLM** ; $\Delta\alpha : 10\%(15\%)$: **ITLM** with $\alpha = \alpha^\star - 10\%(15\%)$. We see significant improvement of **ITLM** over **Baseline** for all the settings.

dataset	MNIST with two-layer CNN							CIFAR-10 with WideResNet16-10		
Systematic Label Error										
$\frac{\#\text{clean}}{\#\text{total}}$	**Baseline**	**ITLM**	**Oracle**	**Centroid**	**1-step**	$\Delta\alpha : 10\%$	$\Delta\alpha : 15\%$	**Baseline**	**ITLM**	**Oracle**
60%	66.69	84.98	92.44	70.25	74.29	85.91	79.80	62.03	81.01	90.14
70%	80.74	89.19	92.82	83.42	84.07	89.76	88.00	73.47	87.08	90.72
80%	89.91	91.93	92.93	90.18	91.38	90.92	89.06	80.17	89.34	91.33
90%	92.35	92.68	93.2	92.44	92.63	91.10	90.62	86.63	90.00	91.74
Random Label Error										
$\frac{\#\text{clean}}{\#\text{total}}$	**Baseline**	**ITLM**	**Oracle**	**Centroid**	**1-step**	$\Delta\alpha : 10\%$	$\Delta\alpha : 15\%$	**Baseline**	**ITLM**	**Oracle**
30%	80.87	84.54	91.37	80.89	93.91	80.39	68.00	49.58	64.74	85.78
50%	88.59	90.16	92.14	88.94	89.13	89.14	86.23	64.74	82.51	89.26
70%	91.18	91.12	92.82	91.25	90.28	90.41	88.37	73.60	88.23	90.72
90%	92.50	92.43	93.20	92.40	92.42	91.48	90.25	86.13	90.33	91.74

6.1. Synthetic experiments

We consider the linear regression setting, where dimension $d = 100$ and sample $n = 1000$, all $\phi(x_i)$s are generated as i.i.d. normal Gaussian vectors. The outputs are generated following (1). The results are based on an average of 100 runs under each setting. The performance for both the random output setting and the mixture model setting are similar, we focus on random output setting in this section. For similar results in mixture model setting, and further results in the general linear setting, please refer to the Appendix.

Results: *(a) Inconsistency.* Figure 2-(a) and (b) show that ITLM gives an inconsistent result, since under the large noise setting, the recovery error does not decrease as sample size increases. Also, (a) suggests that if the noise is small, the final performance is close to the oracle unless sample size is extremely large. These observations match with our theoretic guarantees in Theorem 7 and 8. *(b) Performance v.s. $\boldsymbol{\alpha}^\star$.* Figure 2-(c) shows the recovery result of our algorithm, for both large and small M_t. As $\alpha^\star$ increases, ITLM is able to successfully learn the parameter with high probability. *(c) Convergence rates.* In fact, as implied by Theorem 7, the algorithm has super-linear convergence under the small noise setting. We provide results in Appendix to verify this.

Next, we apply ITLM to many bad data settings with DNN models. We present the result using ITLM with large M since (a) according to Figure 2, both large and small M perform similar in linear setting; (b) full update could be more stable in DNNs, since a set of bad training samples may deceive one gradient update, but is harder to deceive the full training process. Also, we run re-initialization for every round of update to make it harder to stuck at bad local minimum.

6.2. Random/Systematic label error for classification

We demonstrate the effectiveness of ITLM for correcting training label errors in classification by starting from a "clean" dataset, and introducing either one of two different types of errors to make our training and validation data set:

(a) *random errors in labels*: for samples in error, the label is changed to a random incorrect one, independently and with equal probability;
(b) *systematic errors in labels*: for a class "a", all its samples in error are given the *same* wrong label "b"

Intuitively, systematic errors are less benign than random ones since the classifier is given a more directed and hence stronger misleading signal. However, systematic errors can happen when some pairs of classes are more confusable than others. We investigate the ability of ITLM to account for these errors for *(a)* 5% subsampled MNIST (LeCun et al., 1998) [3] dataset with a standard 2-layer CNN, and *(b)* CIFAR-10 (Krizhevsky & Hinton, 2009) with a 16-layer WideResNet (Zagoruyko & Komodakis, 2016). For each of these, **Baseline** represents the standard process of training the (respective) NN model on all the samples. Training details for each dataset are specified in the Appendix. For the CIFAR-10 experiments, we run 4 rounds with early stopping, and then 4 rounds with full training. As motivated

[3] We subsample MNIST by retaining only 5% of its samples, so as to better distinguish the performance of different algorithms. Without this, the MNIST dataset is "too easy" in the sense that it has so many samples that the differences in algorithm performance is muted if all samples are used.

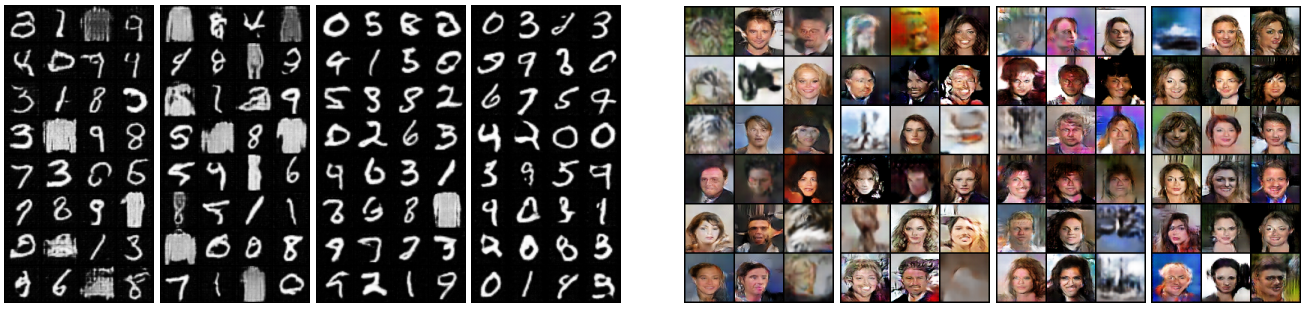


Figure 3. **Qualitative performance of ILFB for GANs:** We apply ILFB to a dataset of (1) **left**: 80% MNIST + 20% Fashion-MNIST; (2) **right**: 70% CelebA + 30% CIFAR-10. The panels show the fake images from 32 randomly chosen (and then fixed) latent vectors, as ILFB iterations update the GAN weights. Baseline is the standard training of fitting to all samples, which generates both types of images, but by the 5th iteration it hones in on digit/face-like images.

in Section 1 (in the main paper), early stopping may help us better filter out bad samples since the later rounds may overfit on them. We set α to be 5% less than the true ratio of clean samples, to simulate the robustness of our method to mis-specified sample ratio. Notice that one can always use cross-validation to find the best α. For the MNIST experiments, we run 5 rounds of ITLM , and we also include comparisons to several heuristic baseline methods, to give more detailed comparison, see the caption thereof.

Results: According to Table 1, we observe significant improvement over the baselines under most settings for MNIST experiments and all settings for CIFAR-10 experiments. ITLM is also not very sensitive to mis-specified clean sample ratio (especially for cleaner dataset). We next compare with two recent state-of-the-art methods focusing on the noisy label problem: (1) MentorNet PD/DD (Jiang et al., 2017) and (2) Reweight (Ren et al., 2018). These methods are based on the idea of curriculum learning and meta-learning. MentorNet DD and Reweight require an additional clean dataset to learn from. As shown in Table 2 (MentorNet results are reported based on their official github page), our method on the 60% clean dataset only drops 6% in accuracy compared with the classifier trained with the original clean dataset, which is the best among the reported results, and is much better than MentorNet PD. For the extremely noisy setting with 20% clean samples, our approach is significantly better than MentorNet PD and close to the performance of MentorNet DD.

6.3. Deep generative models with mixed training data

We consider training a GAN – specifically, the DC-GAN architecture (Radford et al., 2015) – to generate images similar to those from a clean dataset, but when the training data given to it contains some fraction of the samples from a bad dataset. This type of bad data setting may happen very often in practice when the data collector collects a large amount of samples/images, e.g., from web search engine, to learn a generative model, and it could be difficult for the collector to find a rule to filter those incorrect samples. All images are unlabeled, and we do not know which training sample comes from which dataset. We investigated the efficacy of our approach in the following two experiments: A mixture of MNIST (clean) images with Fashion-MNIST (bad) images; A mixture of Celeb-A (clean) face images with CIFAR-10 (bad) object images. We consider different fractions of bad samples in the training data, evaluate their effect on standard GAN training, and then the efficacy of our approach as we execute it for upto 5 iterations.

Table 2. **ITLM** compares with reported state-of-the-art approaches on CIFAR-10. We list their reported numbers for comparison. **Reweight / MentorNet DD** require an additional 1k / 5k clean set of data to learn from (**ITLM** does not require any identified clean data). (acc.-1: accuracy of the algorithm; acc.-2 : accuracy with the full CIFAR-10 dataset.)

method	clean %	acc.-1 / acc.-2	extra clean samples?
1. Ours	60%	86.12 / 92.40	No
2. Reweight	60%	86.92 / 95.5	Yes, 1k
4. MentorNet PD	60%	77.6 / 96	No
5. MentorNet DD	60%	88.7 / 96	Yes, 5k
6. Ours	20%	42.24 / 92.40	No
7. MentorNet PD	20%	28.3 / 96	No
8. MentorNet DD	20%	46.3 / 96	Yes, 5k

We again use the framework of ITLM , while slightly changing the two update steps. Recall that training a GAN consists of updating the weights of both a generator network and a discriminator network; our model parameters $\theta = \{\theta^D, \theta^G\}$ include the parameters of both networks. When selecting samples, we only calculate the discriminator's loss, i.e.,

$$S_t \leftarrow \arg \min_{S:|S|=\lfloor \alpha n \rfloor} \sum_{i \in S} D_{\theta_t^D}(s_i)$$

When updating the parameter, we update both the discriminator and the generator simultaneously, as a regular GAN

Table 3. **Generative models from mixed training data: A quantitative measure** The table depicts the ratio of the clean samples in the training data that are *recovered* by the discriminator when it is run on the training samples. The higher this fraction, the more effective the generator. Our approach shows significant improvements with iteration count.

	MNIST(clean)-Fashion(bad)			CelebA(clean)-CIFAR10(bad)		
orig	90%	80%	70%	90%	80%	70%
iter-1	91.90%	76.84%	77.77%	97.12%	81.34%	75.57%
iter-2	96.05%	91.95%	79.12%	97.33%	88.11%	76.45%
iter-3	99.15%	96.14%	85.66%	97.43%	89.48%	86.63%
iter-4	100.0%	99.67%	91.51%	97.53%	92.89%	82.15%
iter-5	100.0%	100.0%	97.00%	98.14%	92.94%	94.02%

training process. Notice that for different GAN architectures, the loss function for training the discriminator varies, however, we can always find a surrogate loss: the loss of the discriminator for real images. Again, we set α to be 5% less than the true ratio of clean samples.

Results: Figure 3 shows qualitatively how the learned generative model performs as we iteratively learn using ITLM . The generated images only contain digit-type images in the first experiment after the 5th iteration, and similar behavior is observed for the corrupted face image. Table 3 provides a quantitative analysis showing that ITLM selects more and more clean samples iteratively. In Appendix, we also show other simple heuristic methods fail to filter out bad data, and our approach is not sensitive to mis-specified α.

6.4. Defending backdoor attack

Backdoor attack is one of the recent attacking schemes that aims at deceiving DNN classifiers. More specifically, the goal of backdoor attack is by injecting few poisoned samples to the training set, such that the trained DNNs achieve both high performance for the regular testing set and a second testing set created by the adversary whose labels are manipulated. We inject backdoor images using exactly the same process as described in (Tran et al., 2018), i.e., we pick a target class and poison 5% of the target class images as watermarked images from other classes. See Figure 4 for typical samples in a training set. Accordingly, we generate a testing set with all watermarked images, whose labels are all set to the target class. Notice that a regularly trained classifier makes almost perfect prediction in the manipulated testing set. We use ITLM with 4 early stopping rounds and 1 full training round, we set α as 0.98.

Results: Our results on several randomly picked poisoned dataset are shown in Table 4. The algorithm is able to filter out most of the injected samples, and the accuracy on the second testing set achieves zero. The early stopping rounds are very effective in filtering out watermarked samples, whereas without early stopping, the performance is poor.

Table 4. Defending backdoor attack samples, which poisons class a and make them class b. test-1 accuracy refers to the true testing accuracy, while test-2 accuracy refers to the testing accuracy on the test set made by the adversary.

class $a \to b$	shape	naive training test-1 / test-2 acc.	with ITLM test-1 / test-2 acc.
$1 \to 2$	X	90.32 / 97.50	90.31 / 0.10
$9 \to 4$	X	89.83 / 96.30	90.02 / 0.60
$6 \to 0$	L	89.83 / 98.10	89.84 / 1.30
$2 \to 8$	L	90.23 / 97.90	89.70 / 1.20

a-"horse" dataset-1 | b-"horse" (bad) dataset-1 | c-"ship" dataset-2 | d-"ship" (bad) dataset-2

Figure 4. Illustration of typical clean and backdoor samples in backdoor attacked training sets. Shown on the left are a clean "horse" image and a bird image with an 'L'-type watermark around the center from one dataset. Shown on the right are a clean "ship" image and a dog image with an 'X'-type watermark on the right from another dataset.

7. Discussion

We demonstrated the merit of iteratively minimize the trimmed loss, both theoretically for the simpler setting of generalized linear models, and empirically for more challenging ones involving neural networks for classification (with label noise) and GANs (with tainted samples). The ITLM approach is simple, flexible and efficient enough to be applied to most modern machine learning tasks, and can serve as a strong baseline and guide when designing new approaches. It is based on the key observation that when tainted data is present, it helps to look more closely at how the loss of each training sample evolves as the model fitting proceeds. Specifically, and especially early in the model fitting process, tainted samples are seen to have higher loss in a variety of tainted settings. This is also theoretically backed up for the (admittedly simpler) generalized linear model case.

Our paper opens several interesting venues for exploration. It would be good to get a better understanding of why the evolution of loss behaves this way in neural network settings; also when it would not do so. It would also be interesting to characterize theoretically the performance for more cases beyond generalized liner models.

Acknowledgement

We would like to acknowledge NSF grants 1302435 and 1564000 for supporting this research.

References

Balakrishnan, S., Du, S. S., Li, J., and Singh, A. Computationally efficient robust sparse estimation in high dimensions. In *Conference on Learning Theory*, pp. 169–212, 2017a.

Balakrishnan, S., Wainwright, M. J., Yu, B., et al. Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1): 77–120, 2017b.

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

Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., and Srivastava, B. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

Chen, Y., Caramanis, C., and Mannor, S. Robust sparse regression under adversarial corruption. In *International Conference on Machine Learning*, pp. 774–782, 2013.

Čížek, P. General trimmed estimation: robust approach to nonlinear and limited dependent variable models. *Econometric Theory*, 24(6):1500–1529, 2008.

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Steinhardt, J., and Stewart, A. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018.

Frénay, B., Kabán, A., et al. A comprehensive introduction to label noise. In *ESANN*, 2014.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Hössjer, O. Exact computation of the least trimmed squares estimate in simple linear regression. *Computational Statistics & Data Analysis*, 19(3):265–282, 1995.

Huber, P. J. Robust statistics. In *International Encyclopedia of Statistical Science*, pp. 1248–1251. Springer, 2011.

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. *arXiv preprint arXiv:1712.05055*, 2017.

Khetan, A., Lipton, Z. C., and Anandkumar, A. Learning from noisy singly-labeled data. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1sUHgb0Z.

Klivans, A. R., Kothari, P. K., and Meka, R. Efficient algorithms for outlier-robust regression. In *Conference on Learning Theory*, pp. 1420–1430, 2018.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Y. and Liang, Y. Learning mixtures of linear regressions with nearly optimal complexity. In *Conference on Learning Theory*, pp. 1125–1144, 2018.

Liu, K., Dolan-Gavitt, B., and Garg, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. *arXiv preprint arXiv:1805.12185*, 2018a.

Liu, L., Shen, Y., Li, T., and Caramanis, C. High dimensional robust sparse regression. *arXiv preprint arXiv:1805.11643*, 2018b.

Malach, E. and Shalev-Shwartz, S. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pp. 961–971, 2017.

Menon, A. K., Van Rooyen, B., and Natarajan, N. Learning from binary labels with instance-dependent corruption. *arXiv preprint arXiv:1605.00751*, 2016.

Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. On the least trimmed squares estimator. *Algorithmica*, 69(1):148–183, 2014.

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.

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

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Ray, A., Neeman, J., Sanghavi, S., and Shakkottai, S. The search problem in mixture models. *Journal of Machine Learning Research*, 18(206):1–61, 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.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 4331–4340, 2018.

Rousseeuw, P. J. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

Rousseeuw, P. J. and Van Driessen, K. Computing lts regression for large data sets. *Data mining and knowledge discovery*, 12(1):29–45, 2006.

Scott, C., Blanchard, G., and Handy, G. Classification with asymmetric label noise: Consistency and maximal denoising. In *Conference On Learning Theory*, pp. 489–511, 2013.

Sedghi, H., Janzamin, M., and Anandkumar, A. Provable tensor methods for learning mixtures of generalized linear models. In *Artificial Intelligence and Statistics*, pp. 1223–1231, 2016.

Shen, F., Shen, C., van den Hengel, A., and Tang, Z. Approximate least trimmed sum of squares fitting and applications in image analysis. *IEEE Transactions on Image Processing*, 22(5):1836–1847, 2013.

Sukhbaatar, S. and Fergus, R. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2(3):4, 2014.

Tran, B., Li, J., and Madry, A. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pp. 8011–8021, 2018.

Vainsencher, D., Mannor, S., and Xu, H. Ignoring is a bliss: Learning with large noise through reweighting-minimization. In *Conference on Learning Theory*, pp. 1849–1881, 2017.

Vershynin, R. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Víšek, J. Á. The least trimmed squares. part i: Consistency. *Kybernetika*, 42(1):1–36, 2006.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks*, pp. 0. IEEE.

Yang, E., Lozano, A. C., Aravkin, A., et al. A general family of trimmed estimators for robust high-dimensional data analysis. *Electronic Journal of Statistics*, 12(2):3519–3553, 2018.

Yi, X., Caramanis, C., and Sanghavi, S. Alternating minimization for mixed linear regression. In *International Conference on Machine Learning*, pp. 613–621, 2014.

Yi, X., Caramanis, C., and Sanghavi, S. Solving a mixture of many random linear equations by tensor decomposition and alternating minimization. *arXiv preprint arXiv:1608.05749*, 2016.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*, 2016.

Zhang, Z. and Sabuncu, M. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, pp. 8778–8788, 2018.

Zhong, K., Jain, P., and Dhillon, I. S. Mixed linear regression with multiple components. In *Advances in neural information processing systems*, pp. 2190–2198, 2016.