Coresets for Robust Training of Neural Networks against Noisy Labels

Baharan Mirzasoleiman
Department of Computer Science
University of California Los Angeles
Los Angeles, CA
baharan@cs.ucla.edu

Kaidi Cao
Department of Computer Science
Stanford University
Stanford, CA
kaidicao@cs.stanford.edu

Jure Leskovec
Department of Computer Science
Stanford University
Stanford, CA
jure@cs.stanford.edu

Abstract

Modern neural networks have the capacity to overfit noisy labels frequently found in real-world datasets. Although great progress has been made, existing techniques are limited in providing theoretical guarantees for the performance of the neural networks trained with noisy labels. Here we propose a novel approach with strong theoretical guarantees for robust training of neural networks trained with noisy labels. The key idea behind our method is to select subsets of clean data points that provide an approximately low-rank Jacobian matrix. We then prove that gradient descent applied to the subsets cannot overfit the noisy labels, even without regularization or early stopping. Our extensive experiments corroborate our theory and demonstrate that deep networks trained on our subsets achieve a significantly superior performance, e.g., 7% increase in accuracy on mini Webvision with 50% noisy labels, compared to state-of-the-art.

1 Introduction

The success of deep neural networks relies heavily on the quality of training data, and in particular accurate labels of the training examples. However, maintaining label quality becomes very expensive for large datasets, and hence mislabeled data points are ubiquitous in large real-world datasets [21]. As deep neural networks have the capacity to essentially memorize any (even random) labeling of the data [49], noisy labels have a drastic effect on the generalization performance of deep neural networks. Therefore, it becomes crucial to develop methods with strong theoretical guarantees for robust training of neural networks against noisy labels. Such guarantees become of the utmost importance in safety-critical systems, such as aircraft, autonomous cars, and medical devices.

There has been a great empirical progress in robust training of neural networks against noisy labels. Existing directions mainly focus on: estimating the noise transition matrix [13, 34], designing robust loss functions [12, 42, 44, 52], correcting noisy labels [26, 36, 41], using explicit regularization techniques [7, 50, 51], and selecting or reweighting training examples [9, 14, 17, 27, 37, 44]. In general, estimating the noise transition matrix is challenging, correcting noisy labels is vulnerable to overfitting, and designing robust loss functions or using explicit regularization cannot achieve


We conduct experiments on noisy versions of CIFAR-10 and CIFAR-100 [22]. Specifically, Hu et al. [16] proved that when width of the hidden layers is sufficiently large (polynomial in the size of the training data), gradient descent with regularization by distance to initialization corresponds to kernel ridge regression using the Neural Tangent Kernel (NTK). Kernel ridge regression performs comparably to early-stopped gradient descent [35, 45], and leads to a generalization guarantee in presence of noisy labels. In another work, Li et al. [23] proved that under a rich (clusterable) dataset model, a one-hidden layer neural network trained with gradient descent first fits the correct labels, and then starts to overfit the noisy labels. This is consistent with the previous empirical findings showing that deep networks tend to learn simple examples first, then gradually memorize harder instances [5]. In practice, however, regularization and early-stopping provide robustness only under relatively low levels of noise (up to 20% of noisy labels) [16, 23].

Here we develop a principled technique, CRUST, with strong theoretical guarantees for robust training of neural networks against noisy labels. The key idea of our method is to carefully select subsets of clean data points that allow the neural network to effectively learn from the training data, but prevent it to overfit noisy labels. To find such subsets, we rely on recent results that characterize the training dynamics based on properties of neural network Jacobian matrix containing all its first-order partial derivatives. In particular, (1) learning along prominent singular vectors of the Jacobian is fast and generalizes well, while learning along small singular vectors is slow and leads to overfitting; and (2) label noise falls on the space of small singular values and impedes generalization [32]. To effectively and robustly learn from the training data, CRUST efficiently finds subsets of clean and diverse data points for which the neural network has an approximately low-rank Jacobian matrix.

We show that the set of medoids of data points in the gradient space that minimizes the average gradient dissimilarity to all the other data points satisfies the above properties. To avoid overfitting noisy labels, CRUST iteratively extracts and trains on the set of updated medoids. We prove that for large enough coresets and a constant fraction of noisy labels, deep networks trained with gradient descent on the medoids found by CRUST correctly classify all the data points. We then explain how mixing up [51] the centers with a few other data points reduces the error of gradient descent updates on the coresets and further improves the generalization performance.

We conduct experiments on noisy versions of CIFAR-10 and CIFAR-100 [22] with noisy labels generated by random flipping the original ones, and the mini Webvision datasets [24] which is a benchmark consisting of images crawled from websites, containing real-world noisy labels. Empirical results demonstrate that, even under extreme noise (i.e., 50% noisy labels), the robustness of deep models trained by CRUST is superior to state-of-the-art baselines, e.g. 7% increase in accuracy on mini Webvision. We note that CRUST achieves state-of-the-art performance without the need for training any auxiliary model or utilizing an extra clean dataset.

2 Additional Related Work

In practice, deeper and wider neural networks generalize better [40, 48]. Theoretically, recent results proved that when the number of hidden nodes is polynomial in the size of the dataset, neural network parameters stay close to their initialization, where the training landscape is almost linear [1, 4, 8, 11], or convex and semi-smooth [2]. For such networks, (stochastic) gradient descent with random initialization can almost always drive the training loss to 0, and overfit any (random or noisy) labeling of the data. Importantly, these results utilize the property that the Jacobian of the neural network is well-conditioned at a random initialization if the dataset is sufficiently diverse.

More closely related to our work is the recent result of [32] which proved that along the directions associated with large singular values of a neural network Jacobian, learning is fast and generalizes well. In contrast, early stopping can help with generalization along directions associated with small singular values. This is consistent with prior results proving the effectiveness of regularization and

2
early stopping for providing robustness against noisy labels [16, 23]. These results, however, are restricted to unrealistically wide networks, and in practice are only effective under low levels of noise.

On the other hand, our method, CRUST, provides rigorous guarantees for robust training of arbitrary deep neural network against noisy labels, by efficiently extracting subsets of data points that provide a low-rank approximation of the Jacobian matrix during the training. Effectively, the extracted subsets do not let the network to overfit noise, hence CRUST can quickly train a model that generalizes well, without the need for regularization or early-stopping. Unlike existing analytical results that are limited to a neighborhood around a random initialization, our method captures the change in Jacobian structure of deep networks for arbitrary parameter values during training. As a result, it achieves state-of-the-art performance both under mild as well as severe noise.

3 Problem Setting: Learning from Noisy Labeled Data

In this section we formally describe the problem of learning from datasets with noisy labels. Suppose we have a dataset $D = \{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$, where $(x_i, y_i)$ denotes the $i$-th sample with input $x_i \in \mathbb{R}^d$ and its observed label $y_i \in \mathbb{R}$. We assume that the labels $\{y\}_{i=1}^n$ belong to one of $C$ classes. Specifically, $y_i \in \{\nu_1, \nu_2, \ldots, \nu_C\}$ with $\{\nu_j\}_{j=1}^C \in [-1, 1]$. We further assume that the labels are separated with margin $\delta \leq |\nu_r - \nu_s|$ for all $r, s \in [C], r \neq s$. Suppose we only observe inputs and their noisy labels $\{\tilde{y}\}_{i=1}^n$, but do not observe true labels $\{y\}_{i=1}^n$. For each class $1 \leq j \leq C$, at most $\rho$ fraction of the labels associated with that class are assigned to another label chosen from $\{\nu_j\}_{j=1}^C$.

Let $f(W, x)$ be an $L$-layer fully connected neural network with scalar output, where $x \in \mathbb{R}^d$ is the input and $W = (W^{(1)}, \ldots, W^{(L)})$ is all the network parameters. Here, $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix in the $l$-th layer ($d_0 = d, d_L = 1$). For simplicity, we assume all the parameters are aggregated in a vector, i.e., $W \in \mathbb{R}^m$, where $m = \sum_{l=2}^{L} d_l \times d_{l-1}$. Suppose that the network is trained by minimizing the squared loss over the noisy training dataset $D = \{(x_i, y_i)\}_{i=1}^n$:

$$L(W) = \frac{1}{2} \sum_{i \in V} (y_i - f(W, x_i))^2,$$  \hspace{1cm} (1)

where $V = \{1, \ldots, n\}$ is the set of all training examples. We apply gradient descent with a constant learning rate $\eta$, starting from an initial point $W_0$ to minimize $L(W)$. The iterations take the form

$$W_{t+1} = W_t - \eta \nabla L(W_t, X), \hspace{0.5cm} \forall \nabla L(W, X) = J^T(W, X)(f(W, X) - y),$$  \hspace{1cm} (2)

where $J(W, X) \in \mathbb{R}^{n \times m}$ is the Jacobian matrix associated with the nonlinear mapping $f$ defined as

$$J(W, X) = \left[ \frac{\partial f(W, x_1)}{\partial W} \ldots \frac{\partial f(W, x_n)}{\partial W} \right]^T.$$  \hspace{1cm} (3)

The goal is to learn a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ (in the form of a neural network) that can predict the true labels $\{\tilde{y}\}_{i=1}^n$ on the dataset $D$. In the rest of the paper, we use $X = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n)^T, \tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_n)^T$.

4 Our Approach: CRUST

In this section we present our main results. We first introduce our method CRUST that selects subsets of clean data points that has an approximately low-rank Jacobian and do not allow gradient descent to overfit noisy labels. Then, we show how mixing up the subsets with a few other data points can further reduce the error of gradient descent updates, and improve the generalization performance.

4.1 Extracting Clean Subsets with Approximately Low-rank Jacobian

The key idea of our method is to carefully select subsets of data points that allow the neural network to effectively learn from the clean training data, but prevent it to overfit noisy labels. Recent result on optimization and generalization of neural networks show that the Jacobian of typical neural networks exhibits an approximately low-rank structure, i.e., a number of singular values are large and the remaining majority of the spectrum consists of small singular values. Consequently, the Jacobian spectrum can be split into information space $\mathcal{I}$, and nuisance space $\mathcal{N}$, associated with the large and small singular values [32].
There are two key observations [23, 32]: (1) While learning along the low-dimensional information space is fast and generalizes well, learning over the high-dimensional nuisance space is slow and leads to overfitting; and (2) The generalization capability and dynamics of training is dictated by how well the label, \( y \), and residual vector, \( r = f(W, X) - y \), are aligned with the information space. If the residual vector is very well aligned with the singular vectors associated with the top singular values of \( J(W, X) \), the gradient update, \( \nabla L(W) = J^T(W, X) r \), significantly reduces the misfit allowing substantial reduction in the train error. The residual and label of most of the clean data points fall on the information space, while the residual and label of noisy data points fall on the nuisance space and impede training and generalization. Formally, for all unit norm clean residuals \( v \in S_c \subset \mathbb{R}^n \) that lie on the signal subspace \( S_c \) and all unit norm noisy residuals \( w \in S_n \subset \mathbb{R}^n \) that lie on the noise subspace \( S_n \), and scalars \( 0 \leq \mu \ll \alpha \leq \beta \), we have

\[
\alpha = \| J^T(W, X) v \|_2 \leq \beta, \quad \text{and} \quad \| J^T(W, X) w \|_2 \leq \mu.
\]  

To avoid overfitting noisy labels, one can leverage the first observation above, and iteratively selects subsets \( S \) of \( k \) data points that provide the best rank-\( k \) approximation to the Jacobian matrix. In doing so, gradient descent applied to the subsets cannot overfit noisy labels. Formally:

\[
S^*(W) = \arg \min_{S \subset V} \| J(W, X) - J(W, X_S) \|_2 \quad \text{s.t.} \quad |S| \leq k.
\]  

Existing techniques to find the best subset of \( k \) columns from an \( n \times m \) matrix have a computational complexity of \( \text{poly}(n, m, k) \) [3, 10, 20], where \( n, m \) are the number of data points and parameters in the network. The subset \( S^*(W) \) depends on the parameter vector \( W \) and a new subset should be extracted after every parameter update. Furthermore, calculating the Jacobian matrix requires backpropagation on the entire dataset which could be very expensive for deep networks. Therefore, the computational complexity of the above methods becomes prohibitive for over-parameterized neural networks trained on large datasets. Moreover, while this approach prohibits overfitting, it does not help identifying the clean data points.

To achieve a good generalization performance, our approach takes advantage of both above mentioned observations. In particular, our goal is to find representative subsets of \( k \) diverse data points with clean labels that span the information space \( \mathcal{I} \), and provide an approximately low-rank Jacobian matrix. The important observation is that as nuisance space is very high dimensional, data points with noisy labels are (uniformly) spread out in the low-dimensional information space. In contrast, information space is fast and generalizes well, learning over the high-dimensional nuisance space is slow and leads to overfitting.

The set of most centrally located clean data points in the gradient space can be found by solving the following \( k \)-medoids problem:

\[
S^*(W) \in \arg \min_{S \subset V} \sum_{i \in S} \min_{j \in S} d_{ij}(W) \quad \text{s.t.} \quad |S| \leq k,
\]  

where \( d_{ij}(W) = \| \nabla \mathcal{L}(W, x_i) - \nabla \mathcal{L}(W, x_j) \|_2 \) is the pairwise dissimilarity between gradients of data points \( i \) and \( j \). Note that the above formulation does not provide the best rank-\( k \) approximation of the Jacobian matrix. However, as the \( k \)-medoids objective selects a diverse set of clean data points, the minimum singular value of the Jacobian of the selected subset projected over the signal subspace \( \sigma_{\min}(J(W, X_S^*), S_c) \), will be large. Next, we weight the derivative of every medoid \( j \in S^* \) by the size of its corresponding cluster \( r_j = \sum_{i \in V} I[j = \arg \min_{s \in S} d_{is}] \) to create the weighted Jacobian matrix \( J_r(W, X_s^*) = \text{diag}(r_1, \ldots, r_k, 0, \ldots, 0) J(W, X_s^*) \). We can establish the following upper and lower bounds on the singular values \( \sigma_{\min[(k)]}(J_r(W, X_s^*), S_c) \) of the weighted Jacobian projected over \( S_c \):

\[
\sqrt{r_{\min}} \sigma_{\min}(J(W, X_s^*), S_c) \leq \sigma_{\min[k]}(J_r(W, X_s^*), S_c) \leq \sqrt{r_{\max}} \| J(W, X_s^*) \|.
\]  

where \( r_{\min} = \min_{j \in [k]} r_j \) and \( r_{\max} = \max_{j \in [k]} r_j \), and we get an error of \( \epsilon = \sqrt{r_{\max}} \| J(W, X_s^*) \| - \| J(W, X_s^*) \| \). Now, we apply gradient descent updates in Eq. (2) to the weighted Jacobian \( J_r(W, X_s^*) \) of the \( k \) extracted medoids:

\[
W_{t+1} = W_t - \eta J_r^T(W_t, X_s^*) (f(W_t, X_s^*) - y_s^*).
\]  

Note that we still need backpropagation on the entire dataset to be able to compute pairwise dissimilarities \( d_{ij} \). For neural networks, the variation of the gradient norms is mostly captured by the gradient of the loss w.r.t. the input to the last layer of the network [19]. This argument can be used to efficiently upper-bound the normed difference between pairwise gradient dissimilarities [30]:

\[
d_{ij}(W) = \| \nabla \mathcal{L}(W, x_i) - \nabla \mathcal{L}(W, x_j) \|_2 \leq O(\| \Sigma'_{L_i}(z_i^t) \nabla^L_i \mathcal{L} - \Sigma'_{L_j}(z_j^t) \nabla^L_j \mathcal{L} \|_2) = d_{ij}^L(W).
\]
where $\Sigma_i^j(z_i^j)\nabla_i^jL$ is gradient of the loss function $L$ w.r.t. the input to the last layer $L$ for data point $i$. The above upper-bound is marginally more expensive to calculate than the value of the loss since it can be computed in a closed form in terms of $z_i^j$. Hence, $d_{ij}^\eta$ can be efficiently calculated. We note that although the upper-bounds $d_{ij}^\eta$ have a lower dimensionality than $d_{ij}$, noisy data points are still spread out in this lower-dimensional space, and hence are not selected as medoids. Our experiments confirm that medoids of the data points in the gradient space have clean labels (Fig. 1 (a)).

Having upper-bounds on the pairwise gradient dissimilarities, we can efficiently find a near-optimal solution for problem (6) by turning it into a submodular maximization problem. A set function $F : 2^V \rightarrow \mathbb{R}^+$ is submodular if $F(S \cup \{e\}) - F(S) \geq F(T \cup \{e\}) - F(T)$, for any $S \subseteq T \subseteq V$ and $e \in V \setminus T$. $F$ is monotone if $F(S|S) \geq 0$ for any $e \in V \setminus S$ and $S \subseteq V$. Minimizing the objective in Problem (6) is equivalent to maximizing the following submodular facility location function:

$$S^*(W) \in \arg\max_{|S| \leq k} F(S, W), \quad F(S, W) = \sum_{i \in V} \max_{j \in S} d_0 - d_{ij}(W), \quad (10)$$

where $d_0$ is a constant satisfying $d_0 \geq d_{ij}(W)$, for all $i, j \in V$. For maximizing the above monotone submodular function, the classical greedy algorithm provides a constant $(1 - 1/e)$-approximation. The greedy algorithm starts with the empty set $S_0 = \emptyset$, and at each iteration $t$, it chooses an element $e \in V$ that maximizes the marginal utility $F(e|S_t) = F(S_t \cup \{e\}) - F(S_t)$. Formally, $S_t = S_{t-1} \cup (\arg\max_{e \in V} F(e|S_{t-1}))$. The computational complexity of the greedy algorithm is $O(nk)$. However, its complexity can be reduced to $O(|V|)$ using stochastic methods [29], and can be further improved using lazy evaluation [28] and distributed implementations [31]. Note that this complexity does not involve any backpropagation as we use the upper-bounds calculated in Eq. (9).

Hence, the subsets can be found very efficiently, in parallel from all classes. Unlike majority of the existing techniques for robust training against noisy labels that has a large computational complexity, robust training with Crust is even faster than training on the entire dataset. We also note that Problem (10) can be addressed in the streaming scenario for very large datasets [6].

The size of the coresets that provide an $\epsilon$ error in approximating the largest singular value of the Jacobian depends on the structure of the problem. For any specific value of $\epsilon$, the coresets can be greedily amended until the desired $\epsilon$ error is achieved. However, the coresets should not be too large to allow overfitting. In practice, neural networks have a low-rank Jacobian and hence relatively small coresets can closely approximate the Jacobian matrix. In our experiments, we show that coresets of size 50% of the training data size provide a superior performance compared to the baselines under various noisy scenarios (Table 1, 3).

The following theorem guarantees that for a constant fraction $\rho$ of noisy labels, deep networks trained with gradient descent on the coresets found by Crust can correctly classify all the data points.

**Theorem 4.1** Assume that we apply gradient descent on the least-squares loss in Eq. (2) to train a neural network on a dataset with label margin $\delta$ and a fraction of $\rho \leq 6/8$ noisy labels. Furthermore, suppose that the Jacobian mapping is $L$-smooth. If coresets found by Crust approximate the Jacobian matrix by an error of at most $\epsilon \leq O\left(\frac{\delta \alpha^2}{k^3 \log(\frac{\delta}{\epsilon})}\right)$, where $\alpha = \sqrt{\tau_{\min} \sigma_{\max}(\nabla^2)}, \beta = \|\nabla^2(W, X)\| + \epsilon$. Then using step size $\eta = \frac{1}{2\beta^2} \min(1, \frac{\alpha^2}{L\sqrt{n}})$, after $\tau = O\left(\frac{k}{\eta \alpha \sigma} \log(\frac{\sqrt{2n}}{\rho})\right)$ iterations the network classifies all the data points correctly.

The proof can be found in the Appendix.

Next, we discuss how to reduce the error of backpropagation on the weighted centers.

### 4.2 Further Reducing the Error of Coresets

There are two potential sources of error during weighted gradient descent updates in Eq. (8). First, we have an $\epsilon$ error in estimating the prominent singular value of the Jacobian matrix. And second, although the $k$-medoids formulation selects centers of clustered clean data points in the gradient space, there is still a small chance, in particular early in training process when the gradients are more uniformly distributed, that the coresets contain some noisy labels. Both errors can be alleviated if we

---

\(^2\)Note that, if $\frac{\partial J(W, X)}{\partial W}$ is continuous, the smoothness condition holds over any compact domain (albeit for a possibly large $L$).
slightly relax the constraint of training on \textit{exact} feature vectors and their labels, and allow training on combinations of every center \(j \in S^*\) with a few examples in its corresponding cluster \(V_j\).

This is the idea behind mixup \cite{zhang2018}. It extends the training distribution with convex combinations of pairs of examples and their labels. For every cluster \(V_j\), we select a few data points \(R_j \subset V_j \setminus \{j\}, |R_j| \ll |V_j|\) uniformly at random, and for every data point \((x_i, y_i), i \in R_j\), we mix it up with the corresponding center \((x_j, y_j), j \in S^*\), to get the set \(\hat{V}_j\) of mixed up points:

\[
\hat{V}_j = \{(\hat{x}, \hat{y}) \mid \hat{x} = \lambda x_i + (1 - \lambda) x_j, \quad \hat{y} = \lambda y_i + (1 - \lambda) y_j \quad \forall i \in R_j\}, \tag{11}
\]

where \(\lambda \sim \text{Beta}(\alpha, \alpha) \in [0, 1]\) and \(\alpha \in \mathbb{R}^+\). The mixup hyper-parameter \(\alpha\) controls the strength of interpolation between feature-label pairs. Our experiments show that the subsets chosen by \textit{CRUST} contain mostly clean data points, but may contain some noisy labels early in the training (Fig. 1 (a)).Mixing up the centers with as few as one example from the corresponding cluster \(C\) (Fig. 1 (a)). Mixing up the centers with as few as one example from the corresponding cluster \(C\) further helps improving the generalization performance (c.f. Table 2).

4.3 Iteratively Reducing Noise

The subsets \(S^*\) in Problem (10) depend on the parameter vector \(W\) and need to be updated during the training. Let \(W^\tau\) be the parameter vector at iteration \(\tau\). At update time \(\tau \in [T]\), we first classify data points based on the updated predictions \(y^\tau = f(W^\tau, X)\). We denote by \(U^\tau_c = \{(x_i, y_i) \in D \mid y^\tau = \nu_c\}\) the set of data points labeled as \(\nu_c\) in iteration \(\tau\). Then, we find \(S(W^\tau)\) by greedily extracting \(k\)-\(n_c\) medoids from each class, where \(n_c\) is the fraction of data points in cluster \(c\). Finding separate coresets from each class can further help to not cluster together noisy data points spread out in the nuisance space, and improves the accuracy of the extracted coresets. Next, we partition the data points in every class to \(k\)-\(n_c\) parts by assigning every data point to its closest medoid. Formally, for partition \(V_j^\tau\) we have \(V_j^\tau = \{i \in U^\tau_c \mid j = \arg \min_{c \in S(W^\tau)} d_{ij}^{\tau}\}\). Finally, we take a small random sample \(R_j^\tau\) from every partition \(V_j^\tau\), and for every data point \(i \in R_j^\tau\) we mix it up with the corresponding medoid \(j \in S(W^\tau)\) according to Eq. (11), and add the generated set \(\hat{D}_j^\tau\) of mixed up data points to the training set. In our experiments we use \(|R_j^\tau| = 1\). At update time \(\tau\), we train on the union of sets generated by mixup, i.e. \(D^\tau = \{\hat{D}_1^\tau \cup \cdots \cup \hat{D}_k^\tau\}\), where every data point \(i \in \hat{D}_j^\tau\) is weighted by \(r_i = |V_j^\tau|/|R_j^\tau|\). The pseudo code of \textit{CRUST} is given in Algorithm 1.

\begin{algorithm}
\caption{Coresets for Robust Training against Noisy Labels (CRUST)}
\begin{algorithmic}
\State \textbf{Input:} The noisy set \(D = \{(x_i, y_i)\}_{i=1}^n\), number of iterations \(T\).
\State \textbf{Output:} Output model parameters \(W^T\).
\For{\(\tau = 1, \ldots, T\)}
\State \(S^\tau = \varnothing\)
\For{\(c \in \{1, \ldots, C\}\)}
\State \(U^\tau_c = \{(x_i, y_i) \in D \mid f(W^\tau, x_i) = \nu_c\}, n_c = |U^\tau_c|/n\). \hfill \text{\Comment{Classify based on predictions.}}
\State \(d_{ij}^{\tau_u} = \text{upper-bounded pairwise gradient dissimilarities for } i, j \in U^\tau_c\) \hfill \text{\Comment{Eq. 9.}}
\State \(S^\tau_c = \{k\cdot n_c\text{-medoids from }U^\tau_c \text{ using } d_{ij}^{\tau_u}\} \hfill \text{\Comment{The greedy algorithm.}}
\For{\(j \in S^\tau_c\)}
\State \(V^\tau_j = \{i \in U^\tau_c \mid j = \arg \min_{c \in S(W^\tau)} d_{ij}^{\tau_u}\}\)
\State \(R^\tau_j = \text{small random sample from }V^\tau_j\).
\State \(\hat{D}_j^\tau = \text{Mixup }(x_j, y_j)\) with \(\{(x_i, y_i) \mid i \in R^\tau_j\}\) \hfill \text{\Comment{Eq. (11).}}
\State \(r^\tau_i = |V^\tau_j|/|R^\tau_j|, \forall i \in R^\tau_j\) \hfill \text{\Comment{Coreset weights in Eq. (8).}}
\State \(S^\tau = S^\tau \cup \hat{D}_j^\tau\)
\EndFor
\EndFor
\State Update the parameters \(W^\tau\) using weighted gradient descent on \(S^\tau\). \hfill \text{\Comment{Eq. (2).}}
\EndFor
\end{algorithmic}
\end{algorithm}
Table 1: Average test accuracy (5 runs) on CIFAR-10 and CIFAR-100. The best test accuracy is marked in bold. CRUST achieves up to 6% improvement (3.15% in average) over the strongest baseline INCV. We note the superior performance of CRUST under 80% label noise.

<table>
<thead>
<tr>
<th>Noise Type</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>F-correction</td>
<td>85.1 ± 0.4</td>
<td>76.0 ± 0.2</td>
</tr>
<tr>
<td>Decoupling</td>
<td>86.7 ± 0.3</td>
<td>79.3 ± 0.6</td>
</tr>
<tr>
<td>Co-teaching</td>
<td>89.1 ± 0.3</td>
<td>82.1 ± 0.6</td>
</tr>
<tr>
<td>MentorNet</td>
<td>88.4 ± 0.5</td>
<td>77.1 ± 0.4</td>
</tr>
<tr>
<td>T-Revision</td>
<td>86.1 ± 0.4</td>
<td>67.4 ± 3.6</td>
</tr>
<tr>
<td>INCV</td>
<td>89.7 ± 0.2</td>
<td>84.8 ± 0.3</td>
</tr>
<tr>
<td>L_DMI</td>
<td>79.3 ± 0.5</td>
<td>78.5 ± 0.6</td>
</tr>
<tr>
<td>CRUST</td>
<td>91.1 ± 0.2</td>
<td>86.3 ± 0.3</td>
</tr>
</tbody>
</table>

5 Experiments

We evaluate our method on artificially corrupted versions of CIFAR-10 and CIFAR-100 [22] with controllable degrees of label noise, as well as a real-world large-scale dataset mini WebVision [24], which contains real noisy labels. Our algorithm is developed with PyTorch [33]. We use 1 Nvidia GTX 1080 Ti for all CIFAR experiments and 4 for training on the mini WebVision dataset.

Baselines. We compare our approach with multiple state-of-the-art methods for robust training against label corruption. (1) F-correction [34] first naively trains a neural network using ERM, then estimates the noise transition matrix $T$. $T$ is then used to construct a corrected loss function with which the model will be retrained. (2) MentorNet [17] first pretrained a teacher network to mimic a curriculum. The student network is then trained with the sample reweighting scheme provided by the teacher network. (4) D2L [26] reduces the effect of noisy labels on learning the true data distribution after learning rate annealing using corrected labels. (3) Decoupling [27] trains two networks simultaneously, and the two networks only train on a subset of samples that do not have the same prediction in every mini batch. (5) Co-teaching [14] also maintains two networks in the training time. Each network selects clean data (samples with small loss) and guide the other network to train on its selected clean subset. (6) INCV [9] first estimates the noise transition matrix $T$ through cross validation, then applies iterative Co-teaching by including samples with small losses. (7) T-Revision [46] designs a deep-learning-based risk-consistent estimator to tune the transition matrix accurately. (8) L_DMI [47] proposes information theoretic noise-robust loss function based on generalized mutual information.

5.1 Empirical results on artificially corrupted CIFAR

We first evaluate our method on CIFAR-10 and CIFAR-100, which contain 50,000 training images and 10,000 test images of size $32 \times 32$ with 10 and 100 classes, respectively. We follow testing protocol adopted in [9, 14], by considering both symmetric and asymmetric label noise. Specifically, we test noise ratio of 0.2, 0.5, 0.8 for symmetric noise, and 0.4 for asymmetric noise.

In our experiments, we train ResNet-32 [15] for 120 epochs with a minibatch of 128. We use SGD with an initial learning rate of 0.1 and decays at epoch 80, 100 by a factor of 10 to optimize the objective, with a momentum of 0.9 and weight decay of $5 \times 10^{-4}$. We only use simple data augmentation following [15]: we first pad 4 pixels on every side of the image, and then randomly crop a $32 \times 32$ image from the padded image. We flip the image horizontally with a probability of 0.5. For CRUST, we select coreset of size 50% of the size of the dataset unless otherwise stated.

We report top-1 test accuracy of various methods in Table 1. Our proposed method CRUST outperforms all the baselines in terms of average test accuracy. While INCV attempts to find a subset of the training set with heuristics, our theoretically-principled method can successfully distinguish data points with correct labels from those with noisy labels, which results in a clear improvement across all different settings. It can be seen that CRUST achieves a consistent improvement by an average of 3.15%, under various symmetric and asymmetric noisy scenarios, compared to the strongest baseline INCV. Interestingly, CRUST achieves the largest improvement of 6% over INCV, under sever 80%
5.2 Ablation study on each component

Here we investigate the effect of each component of CRUSTand its importance, for training on CIFAR-10 with 20% and 50% symmetric noise. Table 2 summarizes the results.

**Effect of the coreset.** Based on the empirical results, the coreset plays an important role in improving the generalization of the trained networks. It is worthwhile noticing that by greedily finding the coreset based on the gradients, we can outperform INCV already. This clearly corroborate our theory and shows the effectiveness of CRUST in filtering the noise and extracting data points with correct labels, compared to other heuristics. It also confirms our argument that although upper-bounded gradient dissimilarities in Eq. (9) has a much lower dimensionality compared to the exact gradients, noisy data points still spread out in the gradient space. Therefore, CRUST can successfully identify central data points with clean labels in the gradient space.

**Effect of mixup and model update.** As discussed in Sec. 4.2, mixup can reduce the bias of estimating the full gradient with the coreset. Moreover, finding separate coresets from each class can further help filtering noisy labels, and improves the accuracy of the extracted coresets. At the beginning of every epoch, CRUST updates the predictions based on the current model parameters and extract corsets from every class separately. We observed that both components, namely mixup and extracting coresets separately from each class based on the predictions of the model being trained, further improve the generalization and hence the final accuracy.

**Size of the coresets.** Fig. 1 demonstrates training curve for CIFAR-10 with 50% symmetric noise. Fig. 1(a) shows the accuracy of coresets of size 30%, 50%, and 70% selected by CRUST. We observe that for various sizes of coresets, the number of noisy centers decreases over time. Furthermore, the fraction of correct labels in the coresets (label accuracy) decreases when the size of the selected centers increases from 30% to 70%. This demonstrates that CRUST identifies clean data points first. Fig. 1 (b), (c) show the train and test accuracy, when training with CRUST on coresets of various sizes. We can see that coresets of size 30% achieve a lower accuracy as they are too small to accurately estimate the spectrum of the information space and achieve a good generalization performance. Coresets of size 70% achieve a lower training and test accuracy compared to coresets.
Table 3: Test accuracy on mini WebVision. The best test accuracy is marked in bold. CRUST achieves up to 7.16% improvement (6.46% in average) in top-1 accuracy over the strongest baseline INCV.

<table>
<thead>
<tr>
<th>Method</th>
<th>WebVision Top-1</th>
<th>WebVision Top-5</th>
<th>ImageNet Top-1</th>
<th>ImageNet Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-correction</td>
<td>61.12</td>
<td>82.68</td>
<td>57.36</td>
<td>82.36</td>
</tr>
<tr>
<td>Decoupling</td>
<td>62.54</td>
<td>84.74</td>
<td>58.26</td>
<td>82.26</td>
</tr>
<tr>
<td>Co-teaching</td>
<td>63.38</td>
<td>85.20</td>
<td>61.48</td>
<td>84.70</td>
</tr>
<tr>
<td>MentorNet</td>
<td>63.00</td>
<td>81.40</td>
<td>57.80</td>
<td>79.92</td>
</tr>
<tr>
<td>D2L</td>
<td>62.68</td>
<td>84.00</td>
<td>57.80</td>
<td>81.36</td>
</tr>
<tr>
<td>INCV</td>
<td>65.24</td>
<td>85.34</td>
<td>61.60</td>
<td>84.98</td>
</tr>
<tr>
<td><strong>CRUST</strong></td>
<td><strong>72.40</strong></td>
<td><strong>89.56</strong></td>
<td><strong>67.36</strong></td>
<td><strong>87.84</strong></td>
</tr>
</tbody>
</table>

of size 50%. As 50% of the labels are noisy, subsets of size 70% contain at least 20% noisy labels and the model eventually overfits the noise.

5.3 Empirical results on mini WebVision

WebVision is real-world dataset with inherent noisy labels [24]. It contains 2.4 million images crawled from Google and Flickr that share the same 1000 classes from the ImageNet dataset. The noise ratio in classes varies from 0.5% to 88% (Fig. 4 in [24] shows the noise distribution). We follow the setting in [17] and create a mini WebVision that consists of the top 50 classes in the Google subset with 66,000 images. We use both WebVision and ImageNet test sets for testing the performance of the model trained on coresets of size 50% of the data found by CRUST. We train InceptionResNet-v2 [39] for 90 epochs with a starting learning rate of 0.1. We anneal the learning rate at epoch 30 and 60, respectively. We start to train only on the coreset at epoch 60. The coreset size we use is 50%. The results are shown in Table 3. Our method consistently outperforms other baselines, and achieves an average of 5% improvement in the test accuracy, compared to INCV.

6 Conclusion

We proposed a novel approach with strong theoretical guarantees for robust training of neural networks against noisy labels. Our method, CRUST, relies on the following key observation: learning along prominent singular vectors of the Jacobian is fast and generalizes well, while learning along small singular vectors is slow and leads to overfitting. To achieve a good generalization performance and avoid overfitting, CRUST iteratively selects subsets of clean data points that provide an approximately low-rank Jacobian matrix. We proved that for a constant fraction of noisy labels, neural networks trained with gradient descent applied to the subsets found by CRUST correctly classify all the data points. Our extensive experiments demonstrated the effectiveness of our method in providing robustness against noisy labels. In particular, we showed that deep networks trained on the our subsets achieve a significantly superior performance, e.g., 7% increase in accuracy on Webvision with 50% noisy labels, compared to state-of-the-art baselines.

Broader Impact

Deep neural networks achieve impressive results in a wide variety of domains, including vision and speech recognition. The quality of the trained deep models on such datasets increases logarithmically with the size of the data [38]. This improvement, however, is contingent on the availability of reliable and accurate labels. In practice, collecting large high quality datasets is often very expensive and time-consuming. For example, labeling of medical images depends on domain experts and hence is very resource-intensive. In some applications, it necessitate obtaining consensus labels or labels from multiple experts and methods for aggregating those annotations to get the ground truth labels [18]. In some domains, crowd-sourcing methods are used to obtain labels from non-experts. An alternative solution is automated mining of data, e.g., from the Internet by using different image-level tags that can be regarded as labels. These solutions are cheaper and more time-efficient than human annotations, but label noise in such datasets is expected to be higher than in expert-labeled datasets. Noisy labels have a drastic effect on the generalization performance of deep neural networks. This prevents deep networks from being employed in real-world noisy scenarios, in particular in safety critical applications such as aircraft, autonomous cars, and medical devices.

State-of-the-art methods for training deep networks with noisy labels are mostly heuristics and cannot provide theoretical guarantees for the robustness of the trained model in presence of noisy labels.
Failure of such systems can have a drastic effect in sensitive and safety critical applications. Our research provides a principled method for training deep networks on real-world datasets with noisy labels. Our proposed method, CRUST, is based on the recent advances in theoretical understanding of neural networks, and provides theoretical guarantee for the performance of the deep networks trained with noisy labels. We expect our method to have a far-reaching impact in deployment of deep neural networks in real-world systems. We believe our research will be beneficial for deep learning in variety of domains, and do not have any societal or ethical disadvantages.

Acknowledgments

We gratefully acknowledge the support of DARPA under Nos. FA865018C7880 (ASED), N660011924033 (MCS); ARO under Nos. W911NF-16-1-0342 (MURI), W911NF-16-1-0171 (DURIP); NSF under Nos. OAC-1835598 (CINES), OAC-1934578 (HDR), CCF-1918940 (Experiments), IIS-2030477 (RAPID); Stanford Data Science Initiative, Wu Tsai Neurosciences Institute, Chan Zuckerberg Biohub, Amazon, Boeing, JPMorgan Chase, Docomo, Hitachi, JD.com, KDDI, NVIDIA, Dell. J. L. is a Chan Zuckerberg Biohub investigator.

References


[14] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi


[17] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning
data-driven curriculum for very deep neural networks on corrupted labels. In International


[19] Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning

On approximation guarantees for greedy low rank optimization. In International Conference on

[21] Ranjay A Krishna, Kenji Hata, Stephanie Chen, Joshua Kravitz, David A Shamma, Li Fei-Fei,


[23] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stop-


[25] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning
from noisy labels with distillation. In Proceedings of the IEEE International Conference on

[26] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi

[27] Eran Malach and Shai Shalev-Shwartz. Decoupling ”when to update” from ”how to update”. In Advances in Neural Information Processing Systems, pages 960–970, 2017.


[29] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas


[31] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular
maximization: Identifying representative elements in massive data. In Advances in Neural


