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 deep networks trained with noisy labels. The key idea behind our method is to select weighted subsets (coresets) of clean data points that provide an approximately low-rank Jacobian matrix. We then prove that gradient descent applied to the subsets do not overfit the noisy labels. Our extensive experiments corroborate our theory and demonstrate that deep networks trained on our subsets achieve a significantly superior performance compared to state-of-the art, e.g., 6% increase in accuracy on CIFAR-10 with 80% noisy labels, and 7% increase in accuracy on mini Webvision[1].

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

[1] Code available at https://github.com/snap-stanford/crust.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

state-of-the-art performance [16, 23, 51]. Therefore, the most promising methods rely on selecting or reweighting training examples by knowledge distillation from auxiliary models [14, 17, 27], or exploiting an extra clean labelled dataset containing no noisy labels [17, 25, 37, 43, 50]. In practice, training reliable auxiliary models could be challenging, and relying on an extra dataset is restrictive as it requires the training and extra dataset to follow the same distribution. Nevertheless, the major limitation of the state-of-the-art methods is their inability to provide theoretical guarantees for the performance of neural networks trained with noisy labels.

There has been a few recent efforts to theoretically explain the effectiveness of regularization and early stopping in generalization of over-parameterized neural networks trained on noisy labels [16, 23]. 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 do not overfit the noisy labels. 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. Effectively, clean coresets found by CRUST improve the generalization performance by reducing the ratio of noisy labels and their alignment with the space of small singular values.

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 the robustness of deep models trained by CRUST is superior to state-of-the-art baselines, e.g. 6% increase in accuracy on CIFAR-10 with 80% noisy labels, and 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 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 networks against noisy labels, by efficiently extracting subsets of clean data points that provide an approximately low-rank Jacobian matrix during the training. Effectively, the extracted subsets do not allow the network to overfit noise, and hence CRUST can quickly train a model that generalizes well. 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 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$, where $(\boldsymbol{x}_i, y_i)$ denotes the i-th sample with input $\boldsymbol{x}_i \in \mathbb{R}^d$ and its observed label $y_i \in \mathbb{R}$. We assume that the labels $\{y_i\}_{i=1}^n$ belong to one of C classes. Specifically, $y_i \in \{\nu_1, \nu_2, \cdots, \nu_C\}$ 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 $\{y_i\}_{i=1}^n$, but do not observe true labels $\{\tilde{y}_i\}_{i=1}^n$. For each class $1 \leq j \leq C$, a fraction of the labels associated with that class are assigned to another label chosen from $\{\nu_j\}_{j=1}^C$.

Let $f(\boldsymbol{W}, \boldsymbol{x})$ be an L-layer fully connected neural network with scalar output, where $\boldsymbol{x} \in \mathbb{R}^d$ is the input and $\boldsymbol{W} = (\boldsymbol{W}^{(1)}, \cdots, \boldsymbol{W}^{(L)})$ is all the network parameters. Here, $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ 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., $\boldsymbol{W} \in \mathbb{R}^m$, 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 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$:

$$\mathcal{L}(\boldsymbol{W}) = \frac{1}{2} \sum_{i \in V} (y_i - f(\boldsymbol{W}, \boldsymbol{x}_i))^2, \tag{1}$$

where $V = \{1, \cdots, n\}$ is the set of all training examples. We apply gradient descent with a constant learning rate η, starting from an initial point $\boldsymbol{W}^0$ to minimize $\mathcal{L}(\boldsymbol{W})$. The iterations take the form

$$\boldsymbol{W}^{\tau+1} = \boldsymbol{W}^\tau - \eta \nabla \mathcal{L}(\boldsymbol{W}^\tau, \boldsymbol{X}), \quad \nabla \mathcal{L}(\boldsymbol{W}, \boldsymbol{X}) = \mathcal{J}^T(\boldsymbol{W}, \boldsymbol{X})(f(\boldsymbol{W}, \boldsymbol{X}) - \boldsymbol{y}), \tag{2}$$

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

$$\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}) = \Big[\frac{\partial f(\boldsymbol{W}, \boldsymbol{x}_1)}{\partial \boldsymbol{W}} \ \cdots \ \frac{\partial f(\boldsymbol{W}, \boldsymbol{x}_n)}{\partial \boldsymbol{W}} \Big]^T. \tag{3}$$

The goal is to learn a function $f : \mathbb{R}^d \to \mathbb{R}$ (in the form of a neural network) that can predict the true labels $\{\tilde{y}_i\}_{i=1}^n$ on the dataset $\mathcal{D}$. In the rest of the paper, we use $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n), \boldsymbol{y} = (y_1, \cdots, y_n)^T, \tilde{\boldsymbol{y}} = (\tilde{y}_1, \cdots, \tilde{y}_n)^T$. Furthermore, we use $\boldsymbol{X}_S$, $\boldsymbol{y}_S$, and $\mathcal{J}(\boldsymbol{W}, X_S)$ to denote inputs, labels, and the Jacobian matrix associated with elements in a subset $S \subseteq V$ of data points, respectively.

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]. Formally, for complementary subspaces $\mathcal{S}_-, \mathcal{S}_+ \subset \mathbb{R}^n$, and for all unit norm vectors $\boldsymbol{v} \in \mathcal{S}_+$, $\boldsymbol{w} \in \mathcal{S}_-$, and scalars $0 \leq \mu \ll \alpha \leq \beta$, we have

$$\alpha \leq \|\mathcal{J}^T(\boldsymbol{W}, \boldsymbol{X})\boldsymbol{v}\|_2 \leq \beta, \quad \text{and} \quad \|\mathcal{J}^T(\boldsymbol{W}, \boldsymbol{X})\boldsymbol{w}\|_2 \leq \mu. \tag{4}$$

There are two key observations [23, 32]: (1) While learning over 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, $\boldsymbol{y}$, and residual vector, $\boldsymbol{r} = f(\boldsymbol{W}, \boldsymbol{X}) - \boldsymbol{y}$, 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 $\mathcal{J}(\boldsymbol{W}, \boldsymbol{X})$, the gradient update, $\nabla\mathcal{L}(\boldsymbol{W}) = \mathcal{J}^T(\boldsymbol{W}, \boldsymbol{X})\boldsymbol{r}$, significantly reduces the misfit allowing substantial reduction in the training error. Importantly, 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.

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 the noisy labels. Formally:

$$S^*(\boldsymbol{W}) = \arg\min_{S \subseteq V} \|\mathcal{J}^T(\boldsymbol{W}, \boldsymbol{X}) - P_S\mathcal{J}^T(\boldsymbol{W}, \boldsymbol{X})\|_2 \quad \text{s.t.} \quad |S| \leq k, \tag{5}$$

where $\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S) \in \mathbb{R}^{k \times m}$ is the set of k rows of the Jacobian matrix associated to $\boldsymbol{X}_S$, and $P_S = \mathcal{J}^T(\boldsymbol{W}, \boldsymbol{X}_S)\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S)$ denotes the projection onto the k-dimensional space spanned by the rows of $\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S)$. Existing techniques to find the best subset of k rows or columns from an $n \times m$ matrix have a computational complexity of $poly(n, m, k)$ [3, 10, 20], where n, m are the number of data points and parameters in the network. Note that the subset $S^*(\boldsymbol{W})$ depends on the parameter vector $\boldsymbol{W}$ 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. Most importantly, 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 the 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 spread out in the *gradient* space. In contrast, information space is low-dimensional and data points with clean labels that have similar gradients cluster closely together. The set of most centrally located clean data points in the gradient space can be found by solving the following k-medoids problem:

$$S^*(\boldsymbol{W}) \in \arg\min_{S \subseteq V} \sum_{i \in V} \min_{j \in S} d_{ij}(\boldsymbol{W}) \qquad \text{s.t.} \quad |S| \leq k, \tag{6}$$

where $d_{ij}(\boldsymbol{W}) = \|\nabla\mathcal{L}(\boldsymbol{W}, \boldsymbol{x}_i) - \nabla\mathcal{L}(\boldsymbol{W}, \boldsymbol{x}_j)\|_2$ 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 subspace $\mathcal{S}_+$, i.e., $\sigma_{\min}(\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}^*_S), \mathcal{S}_+)$, will be large. Next, we weight the derivative of every medoid $j \in S^*$ by the size of its corresponding cluster $r_j = \sum_{i \in V} \mathbb{I}[j = \arg\min_{s \in S^*} d_{is}]$ to create the weighted Jacobian matrix $\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_{S^*}) = \text{diag}([r_1, \cdots, r_k])J(\boldsymbol{W}, \boldsymbol{X}_{S^*}) \in \mathbb{R}^{k \times m}$. We can establish the following upper and lower bounds on the singular values $\sigma_{i \in [k]}(\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_{S^*}), \mathcal{S}_+)$ of the weighted Jacobian over $\mathcal{S}_+$:

$$\sqrt{r_{\min}}\sigma_{\min}(\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_{S^*}), \mathcal{S}_+) \leq \sigma_{i \in [k]}(\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_{S^*}), \mathcal{S}_+) \leq \sqrt{r_{\max}}\|\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_{S^*})\|, \tag{7}$$

where $r_{\min} = \min_{j \in [k]} r_j$ and $r_{\max} = \max_{j \in [k]} r_j$, and we get an error of ϵ in approximating the largest singular value of the neural network Jacobian, $\epsilon \leq |\sqrt{r_{\max}}\|\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_{S^*})\| - \|\mathcal{J}(\boldsymbol{W}, \boldsymbol{X})\||$. Now, we apply gradient descent updates in Eq. (2) to the weighted Jacobian $\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_{S^*})$ of the k extracted medoids:

$$\boldsymbol{W}^{\tau+1} = \boldsymbol{W}^{\tau} - \eta\mathcal{J}_r^T(\boldsymbol{W}, \boldsymbol{X}_{S^*})(f(\boldsymbol{W}, \boldsymbol{X}_{S^*}) - \boldsymbol{y}_{S^*}), \tag{8}$$

Note that we still need backpropagation on the entire dataset to be able to compute pairwise dissimilarities d_{ij}. For neural networks, it is shown that 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]:

$$d_{ij}(\boldsymbol{W}) = \|\nabla\mathcal{L}(\boldsymbol{W}, \boldsymbol{x}_i) - \nabla\mathcal{L}(\boldsymbol{W}, \boldsymbol{x}_j)\|_2 \leq c_1\|\Sigma'_L(\boldsymbol{z}_i^L)\nabla_i^L\mathcal{L} - \Sigma'_L(\boldsymbol{z}_j^L)\nabla_j^L\mathcal{L}\|_2 + c_2, \qquad (9)$$

where $\Sigma'_L(\boldsymbol{z}_i^L)\nabla_i^L\mathcal{L}$ is gradient of the loss function $\mathcal{L}$ w.r.t. the input to the last layer L for data point i, and c_2, c_2 are constants. 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^L. Hence, $d_{ij}^u = \|\Sigma'_L(\boldsymbol{z}_i^L)\nabla_i^L\mathcal{L} - \Sigma'_L(\boldsymbol{z}_j^L)\nabla_j^L\mathcal{L}\|_2$ can be efficiently calculated. We note that although the upper-bounds d_{ij}^u have a lower dimensionality than d_{ij}, noisy data points still spread out in this lower-dimensional space, and hence are not selected as medoids. This is confirmed by out experiments (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 $F : 2^V \to \mathbb{R}^+$ 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 $F(e|S) \geq 0$ 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:

$$S^*(\boldsymbol{W}) \in \arg\max_{\substack{S\subseteq V,\\ |S|\leq k}} F(S, \boldsymbol{W}), \qquad F(S, \boldsymbol{W}) = \sum_{i\in V}\max_{j\in S} d_0 - d'_{ij}(\boldsymbol{W}), \qquad (10)$$

where d_0 is a constant satisfying $d_0 \geq d_{ij}^u(\boldsymbol{W})$, 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 $S_0 = \varnothing$, 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 $\mathcal{O}(nk)$. However, its complexity can be reduced to $\mathcal{O}(|V|)$ 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].

Our experiments confirm that CRUST can successfully find almost all the clean data points (*c.f.* Fig. 1 (a)). The following theorem guarantees that for a small fraction ρ of noisy labels in the selected subsets, deep networks trained with gradient descent do not overfit the noisy labels.

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 noisy labels. Furthermore, suppose that the Jacobian mapping is L-smooth[2]. Assume that the dataset has a label margin of δ, and coresets found by* CRUST *contain a fraction of $\rho < \delta/8$ noisy labels. If the coresets approximate the Jacobian matrix by an error of at most $\epsilon \leq \mathcal{O}(\frac{\delta\alpha^2}{k\beta\log(\sqrt{k}/\rho)})$, where $\alpha = \sqrt{r_{\min}}\sigma_{\min}(\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S))$, $\beta = \|\mathcal{J}(\boldsymbol{W}, \boldsymbol{X})\| + \epsilon$, then for $L \leq \frac{\alpha\beta}{L\sqrt{2k}}$ and step size $\eta = \frac{1}{2\beta^2}$, after $\tau \geq \mathcal{O}(\frac{1}{\eta\alpha^2}\log(\frac{\sqrt{n}}{\rho}))$ iterations the network classifies all the selected elements correctly.*

The proof can be found in the Appendix. Note that the elements of the selected subsets are mostly clean, and hence the noise ratio ρ is much smaller in the subsets compared to the entire dataset. Very recently, [32] showed that the classification error of neural networks trained on noisy datasets of size n is controlled by the portion of the labels that fall over the nuisance space, i.e., $\|\Pi_{\mathcal{N}}(\boldsymbol{y})\|/\sqrt{n}$. Coresets of size k selected by CRUST are mostly clean. For such subsets, the label vector is mostly aligned with the information space, and thus $\|\Pi_{\mathcal{N}}(\boldsymbol{y}_S)\|/\sqrt{k}$ is smaller. Our method improves the generalization performance by extracting subsets S of size k for which $\|\Pi_{\mathcal{N}}(\boldsymbol{y}_S)\|/\sqrt{k} \leq \|\Pi_{\mathcal{N}}(\boldsymbol{y})\|/\sqrt{n}$. While defer the formal generalization proof to future work, our experiments show that even under severe noise (80% noisy labels), CRUST successfully finds the clean data points and achieves a superior generalization performance (*c.f.* Fig. 1).

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

[2]Note that, if $\frac{\partial\mathcal{J}(\boldsymbol{W},\boldsymbol{X})}{\partial\boldsymbol{W}}$ is continuous, the smoothness condition holds over any compact domain (albeit for a possibly large L).

Algorithm 1 CORESETS FOR ROBUST TRAINING AGAINST NOISY LABELS (CRUST)

Input: The noisy set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, number of iterations T.
Output: Output model parameters $\boldsymbol{W}^T$.

for $\tau = 1, \cdots, T$ **do**
  $S^\tau = \varnothing$.
  for $c \in \{1, \cdots, C\}$ **do**
    $U_c^\tau = \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} | f(\boldsymbol{W}^\tau, \boldsymbol{x}_i) = \nu_c\}$, $n_c = |U_c^\tau|/n$. ▷ Classify based on predictions.
    d_{ij}^u =upper-bounded pairwise gradient dissimilarities for $i, j \in U_c^\tau$ ▷ Eq. 9.
    $S_c^\tau = \{k{\cdot}n_c$-medoids from U_c^τ using $d_{ij}^u\}$ ▷ The greedy algorithm.
    for $j \in S_c^\tau$ **do**
      $V_j^\tau = \{i \in U_c^\tau | j = \arg\min_{v \in S_c^\tau} d_{iv}^u\}$
      R_j^τ = small random sample from V_j^τ.
      $\hat{D}_j^\tau$ =Mixup $(\boldsymbol{x}_j, y_j)$ with $\{(\boldsymbol{x}_i, y_i) \mid i \in R_j^\tau\}$ ▷ Eq. (11).
      $r_i = |V_j^\tau|/|R_j^\tau|, \ \forall i \in R_j^\tau$ ▷ Coreset weights in Eq. (8).
      $S^\tau = S^\tau \cup \hat{D}_j^\tau$
    end for
  end for
  Update the parameters $\boldsymbol{W}^\tau$ using weighted gradient descent on S^τ. ▷ Eq. (2).
end for

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 ϵ 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 slightly relax the constraint of training on *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 [51]. 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 $(\boldsymbol{x}_i, y_i), i \in R_j$, we mix it up with the corresponding center $(\boldsymbol{x}_j, y_j), j \in S^*$, to get the set $\hat{D}_j$ of mixed up points:

$$\hat{D}_j = \{(\hat{\boldsymbol{x}}, \hat{y}) \ \mid \ \hat{x} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \ \ \hat{y} = \lambda y_i + (1-\lambda) y_j \quad \forall i \in R_j\}, \tag{11}$$

where $\lambda \sim \text{Beta}(\alpha, \alpha) \in [0, 1]$ and $\alpha \in \mathbb{R}^+$. The mixup hyper-parameter α controls the strength of interpolation between feature-label pairs. Our experiments show that the subsets chosen by CRUST contain mostly clean data points, but may contain some noisy labels early during the training (Fig. 1 (a)). Mixing up the centers with as few as one example from the corresponding cluster, i.e. $|R_j| = 1$, can reduce the effect of potential noisy labels in the selected subsets. Therefore, mixup can further help improving the generalization performance, as is confirmed by our experiments (Table 2).

4.3 Iteratively Reducing Noise

The subsets found in Problem (10) depend on the parameter vector $\boldsymbol{W}$ and need to be updated during the training. Let $\boldsymbol{W}^\tau$ be the parameter vector at iteration τ. At update time $\tau \in [T]$, we first classify data points based on the updated predictions $\boldsymbol{y}^\tau = f(\boldsymbol{W}^\tau, \boldsymbol{X})$. We denote by $U_c^\tau = \{(\boldsymbol{x}_i, y_i) \in \mathcal{D} | f(\boldsymbol{W}^\tau, \boldsymbol{x}_i) = \nu_c\}$ the set of data points labeled as ν_c in iteration τ. Then, we find $S(\boldsymbol{W}^\tau)$ by greedily extracting $k{\cdot}n_c$ medoids from each class, where $n_c = |U_c|/n$ is the fraction of data points in class $c \in [C]$. 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 by assigning every data point to its closest medoid. Formally, for partition V_j^τ we have $V_j^\tau = \{i \in U_c^\tau | j = \arg\min_{j \in S^\tau} d_{ij}^u\}$. Finally, we take a small random sample R_j^τ from every partition V_j^τ, and for every data point $i \in R_j^\tau$ we mix it up with the corresponding medoid $j \in S(\boldsymbol{W}^\tau)$ 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 τ,

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.

Dataset	CIFAR-10				CIFAR-100		
Noise Type	Sym			Asym	Sym		Asym
Noise Ratio	20	50	80	40	20	50	40
F-correction	85.1 ± 0.4	76.0 ± 0.2	34.8 ± 4.5	83.6 ± 2.2	55.8 ± 0.5	43.3 ± 0.7	42.3 ± 0.7
Decoupling	86.7 ± 0.3	79.3 ± 0.6	36.9 ± 4.6	75.3 ± 0.8	57.6 ± 0.5	45.7 ± 0.4	43.1 ± 0.4
Co-teaching	89.1 ± 0.3	82.1 ± 0.6	16.2 ± 3.2	84.6 ± 2.8	64.0 ± 0.3	52.3 ± 0.4	47.7 ± 1.2
MentorNet	88.4 ± 0.5	77.1 ± 0.4	28.9 ± 2.3	77.3 ± 0.8	63.0 ± 0.4	46.4 ± 0.4	42.4 ± 0.5
D2L	86.1 ± 0.4	67.4 ± 3.6	10.0 ± 0.1	85.6 ± 1.2	12.5 ± 4.2	5.6 ± 5.4	14.1 ± 5.8
INCV	89.7 ± 0.2	84.8 ± 0.3	52.3 ± 3.5	86.0 ± 0.5	60.2 ± 0.2	53.1 ± 0.4	50.7 ± 0.2
T-Revision	79.3 ± 0.5	78.5 ± 0.6	36.2 ± 1.6	76.3 ± 0.8	52.4 ± 0.3	37.6 ± 0.3	32.3 ± 0.4
L_DMI	84.3 ± 0.4	78.8 ± 0.5	20.9 ± 2.2	84.8 ± 0.7	56.8 ± 0.4	42.2 ± 0.5	39.5 ± 0.4
CRUST	**91.1 ± 0.2**	**86.3 ± 0.3**	**58.3 ± 1.8**	**88.8 ± 0.4**	**65.2 ± 0.2**	**56.4 ± 0.4**	**53.0 ± 0.2**

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 CRUST is given in Algorithm 1.

Note that while CRUST updates the coreset during the training, as almost all the clean data points are contained in the coresets in every iteration, gradient descent can successfully contract their residuals in Theorem 4.1 and fit the correct labels. Since CRUST finds a new subset at every iteration τ, we need to use $\alpha = \min_\tau \sqrt{r_{\min}^\tau}\sigma_{\min}(\mathcal{J}(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}))$, and $\beta = \max_\tau \sqrt{r_{\max}^\tau}\|\mathcal{J}(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau})\|$ in Theorem 4.1, where α and β are the minimum and maximum singular values of the Jacobian of the subsets weighted by $\boldsymbol{r}^\tau$ found by CRUST, during T steps of gradient descent updates.

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 pretrains 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 32×32 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

Table 2: Ablation study on CIFAR-10 with 20% and 50% symmetric noise. ✓ indicates the corresponding component. coreset w/label and coreset w/label correspond to finding coresets separately from every class based on their observed noisy labels, or labels predicted by the model being trained.

Component				Noise Ratio	
coreset w/ label	coreset w/ pred.	w/o mixup	w/ mixup	20	50
✓		✓		90.21	84.92
✓			✓	90.48	85.23
	✓	✓		90.71	85.57
	✓		✓	91.12	86.27

the objective, with a momentum of 0.9 and weight decay of 5×10^{-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 32×32 image from the padded image. We flip the image horizontally with a probability of 0.5. For CRUST, we select coresets 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% label noise. This shows the effectiveness of our method in extracting data points with clean labels from a large number of noisy data points, compared to other baselines.

5.2 Ablation study on each component

Here we investigate the effect of each component of CRUST and its importance, for robust 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 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.

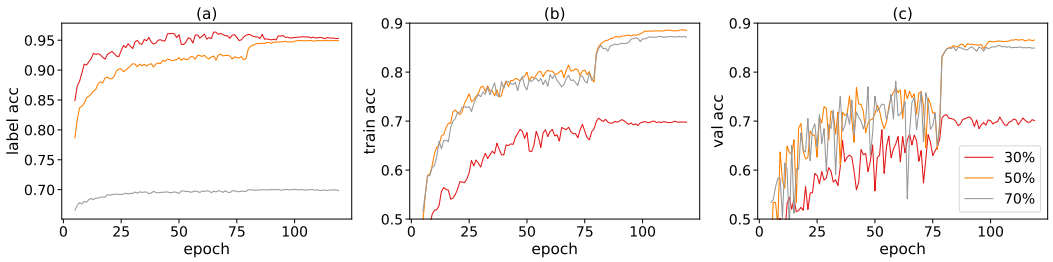

Figure 1: Training curve for CIFAR-10 with 50% symmetric noise. (a) The fraction of correct labels in the coreset (label accuracy) with respect to epochs. (b) The accuracy on the whole training set with respect to epochs. (c) The accuracy on test set with respect to epochs. Here we consider coresets of size 30%, 50%, and 70% of the CIFAR-10 training dataset.

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.

Method	WebVision		ImageNet	
	Top-1	Top-5	Top-1	Top-5
F-correction	61.12	82.68	57.36	82.36
Decoupling	62.54	84.74	58.26	82.26
Co-teaching	63.58	85.20	61.48	84.70
MentorNet	63.00	81.40	57.80	79.92
D2L	62.68	84.00	57.80	81.36
INCV	65.24	85.34	61.60	84.98
CRUST	**72.40**	**89.56**	**67.36**	**87.84**

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. The results are shown in Table 3. It can be seen that 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 observations: (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) The generalization capability and dynamics of training is dictated by how well the label and residual vector are aligned with the information space. 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 in the subsets, neural networks trained with gradient descent applied to the subsets found by CRUST correctly classify all its data points. At the same time, our method improves the generalization performance of the deep network by decreasing the portion of noisy labels that fall over the nuisance space of the network Jacobian. 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., 6% increase in accuracy on CIFAR-10 with 80% noisy labels, and 7% increase in accuracy on mini Webvision, 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 (Expeditions), 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

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252, 2019.

[3] Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. Greedy column subset selection: New bounds and distributed algorithms. In *International Conference on Machine Learning*, pages 2539–2548, 2016.

[4] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332, 2019.

[5] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017.

[6] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings*

of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 671–680, 2014.

[7] Kaidi Cao, Yining Chen, Junwei Lu, Nikos Arechiga, Adrien Gaidon, and Tengyu Ma. Heteroskedastic and imbalanced deep learning with adaptive regularization. *arXiv preprint arXiv:2006.15766*, 2020.

[8] Yuan Cao and Quanquan Gu. A generalization theory of gradient descent for learning over-parameterized deep relu networks. *arXiv preprint arXiv:1902.01384*, 2019.

[9] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pages 1062–1070, 2019.

[10] Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[11] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685, 2019.

[12] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[13] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016.

[14] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pages 8527–8537, 2018.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Wei Hu, Zhiyuan Li, and Dingli Yu. Understanding generalization of deep neural networks trained with noisy labels. *arXiv preprint arXiv:1905.11368*, 2019.

[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 Conference on Machine Learning*, pages 2309–2318, 2018.

[18] Davood Karimi, Haoran Dou, Simon K Warfield, and Ali Gholipour. Deep learning with noisy labels: exploring techniques and remedies in medical image analysis. *arXiv preprint arXiv:1912.02911*, 2019.

[19] Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International Conference on Machine Learning*, pages 2525–2534, 2018.

[20] Rajiv Khanna, Ethan Elenberg, Alexandros Dimakis, Joydeep Ghosh, and Sahand Negahban. On approximation guarantees for greedy low rank optimization. In *International Conference on Machine Learning*, 2017.

[21] Ranjay A Krishna, Kenji Hata, Stephanie Chen, Joshua Kravitz, David A Shamma, Li Fei-Fei, and Michael S Bernstein. Embracing error to enable rapid crowdsourcing. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 3167–3179, 2016.

[22] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[23] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. *arXiv preprint arXiv:1903.11680*, 2019.

[24] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[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 Computer Vision*, pages 1910–1918, 2017.

[26] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, pages 3355–3364, 2018.

[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.

[28] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978.

[29] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[30] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. *arXiv preprint arXiv:1906.01827*, 2019.

[31] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

[32] Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 2020.

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[34] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017.

[35] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1):335–366, 2014.

[36] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[37] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343, 2018.

[38] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[39] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[41] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[42] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems*, pages 10–18, 2015.

[43] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 839–847, 2017.

[44] Xinshao Wang, Yang Hua, Elyor Kodirov, and Neil M Robertson. Imae for noise-robust learning: Mean absolute error does not treat examples equally and gradient magnitude's variance matters. *arXiv preprint arXiv:1903.12141*, 2019.

[45] Yuting Wei, Fanny Yang, and Martin J Wainwright. Early stopping for kernel boosting algorithms: A general analysis with localized complexities. In *Advances in Neural Information Processing Systems*, pages 6065–6075, 2017.

[46] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In *Advances in Neural Information Processing Systems*, pages 6838–6849, 2019.

[47] Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise. In *Advances in Neural Information Processing Systems*, pages 6225–6236, 2019.

[48] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[49] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[50] Han Zhang, Honglak Lee, Sercan Arik, Tomas Pfister, and Zizhao Zhang. Distilling effective supervision from severe label noise. 2020.

[51] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[52] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018.

Appendix

Our main contribution is to show that for any dataset, clean data points cluster together in the gradient space and hence medoids of the gradients (1) have clean labels, and (2) provide a low-rank approximation of the Jacobian, $\mathcal{J}$, of an arbitrary deep network. Hence, training on the medoids is robust to noisy labels. Our analysis of the residuals during gradient descent builds on the analysis of [23], but generalize it to arbitrary deep networks without the need for over-parameterization, and random initialization. Crucially, [23] relies on the following assumptions to show that gradient descent with early stopping is robust to label noise, with a high probability: (1) data $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ has K clusters, (2) neural net f has one hidden layer with k neurons, i.e., $f = \phi(\boldsymbol{X}\boldsymbol{W}^T)\boldsymbol{\nu}$, (3) output weights $\boldsymbol{\nu}$ are fixed to half $+1/\sqrt{k}$, and half $-1/\sqrt{k}$, (4) network is over-parameterized, i.e., $k \geq K^4$, where $K = \mathcal{O}(n)$, (5) the input-to-hidden weights $\boldsymbol{W}^0$ have random Gaussian initialization. Indeed, from the clusterable data assumption it easily follows that the neural network covariance $\Sigma(\boldsymbol{X}) = \mathbb{E}\big[\big(\phi'(\boldsymbol{X}\boldsymbol{W}^T)\phi'(\boldsymbol{W}^T\boldsymbol{X})\big) \odot \big(\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\big)\big] = \frac{1}{k}\mathbb{E}_{\boldsymbol{W}^0}\big[\mathcal{J}(\boldsymbol{W}^0)\mathcal{J}^T(\boldsymbol{W}^0)\big]$ is low-rank, and hence early stopping prevents overfitting noisy labels. In contrast, our results holds for arbitrary deep nets without relying on the above assumptions.

A Proofs for Theorems

The following Corollary builds upon the meta Theorem 7.2 from [23] and captures the contraction of the residuals during gradient descent updates on a dataset with corrupted labels, when the Jacobian is exactly low-rank. The original theorem in [23] is based on the assumptions that the fraction of corrupted labels in the dataset is small, and the dataset is clusterable. Hence $\mathcal{S}_+$ is dictated by the membership of data points to different clusters. In contrast, our method CRUST applies gradient descent only to the selected subsets. Note that the elements of the selected subsets are mostly clean, and hence the extracted subsets have a significantly smaller fraction of noisy labels. The elements selected earlier by CRUST are medoids of the main clusters in the gradient space. As we keep selecting new data points, we extract medoids of the smaller groups of data points within the main clusters. Therefore, in our method the support subspace $\mathcal{S}_+$ is defined by the assignment of the elements of the extracted subsets to the main clusters.

More specifically, assume that there are $K < k$ main clusters in the gradient space. We denote the set of central elements of the main clusters by $\bar{S}$. For a subset $S \subseteq V$ of size k selected by CRUST and upper-bounds d^u on gradient dissimilarities from Eq. (9), let $\Lambda_\ell = \{i \in [k] | \ell = \arg\min_{s \in \bar{S}} d^u_{is}\}$ be the set of elements in S that are closest to an element $\ell \in \bar{S}$, i.e., they lie within the main cluster l in the gradient space. Then, $\mathcal{S}_+$ is characterized by

$$\mathcal{S}_+ = \{\boldsymbol{v} \in \mathbb{R}^k \;\big|\; \boldsymbol{v}_{i_1} = \boldsymbol{v}_{i_2} \quad \text{for all} \quad i_1, i_2 \in \Lambda_\ell \quad \text{and for all } 1 \leq \ell \leq K\}. \tag{12}$$

The following corollary captures the contraction of the residuals during gradient descent on subsets found by CRUST, when the Jacobian is low-rank. In Theorem 4.1, we characterize the contraction of the residuals, when the Jacobian is approximately low-rank, i.e., over $\mathcal{S}_-$ the spectral norm is small but nonzero.

Corollary A.1 *Consider a nonlinear least squares problem of the form* $\mathcal{L}(\boldsymbol{W}, \boldsymbol{X}) = \frac{1}{2}\|f(\boldsymbol{W}, \boldsymbol{X}) - \boldsymbol{y})\|^2_{\ell_2}$. *Assume that the weighted Jacobian of the subset S found by* CRUST *is low-rank, i.e., for all $\boldsymbol{v} \in \mathcal{S}_+$ and $\boldsymbol{w} \in \mathcal{S}_-$ with unit Euclidean norm, and $\beta \geq \alpha > 0$ we have that* $\alpha \leq \big\|\mathcal{J}_r^T(\boldsymbol{W}, X_S)\boldsymbol{v}\big\|_{\ell_2} \leq \beta$ *and* $\big\|\mathcal{J}_r^T(\boldsymbol{W}, X_S)\boldsymbol{w}\big\|_{\ell_2} = 0$. *Moreover, assume that the Jacobian mapping $\mathcal{J}(\boldsymbol{W}, \boldsymbol{X_S})$ associated to the nonlinear mapping f is L-smooth, i.e., for all $\boldsymbol{W}^1, \boldsymbol{W}^2 \in \mathbb{R}^m$ we have* $\big\|\mathcal{J}(\boldsymbol{W}^2) - \mathcal{J}(\boldsymbol{W}^1)\big\| \leq L\big\|\boldsymbol{W}^2 - \boldsymbol{W}^1\big\|_{\ell_2}$.[3] *Also let $\boldsymbol{y}_s, \widetilde{\boldsymbol{y}}_S, \boldsymbol{e} = \boldsymbol{y}_S - \widetilde{\boldsymbol{y}}_S \in \mathbb{R}^k$ denote the corrupted and uncorrupted labels associated with the selected subset, and label corruption, respectively. Furthermore, suppose the initial residual $f(\boldsymbol{W}^0, X_S) - \widetilde{\boldsymbol{y}}_S$ with respect to the uncorrupted labels obey $f(\boldsymbol{W}^0, X_S) - \widetilde{\boldsymbol{y}}_S \in \mathcal{S}_+$. Then, iterates of gradient descent updates of the from (2) with a*

[3] Note that, if $\frac{\partial \mathcal{J}(\boldsymbol{W})}{\partial \boldsymbol{W}}$ is continuous, the smoothness condition holds over any compact domain (albeit for a possibly large L).

learning rate $\eta \leq \frac{1}{2\beta^2} \min\left(1, \frac{\alpha\beta}{L\|\boldsymbol{r}^0\|_{\ell_2}}\right)$, *obey*

$$\|\boldsymbol{W}^\tau - \boldsymbol{W}^0\|_{\ell_2} \leq \frac{4\|\boldsymbol{r}^0\|_{\ell_2}}{\alpha} = \frac{4\|f(\boldsymbol{W}^0, X_S) - \boldsymbol{y}_S\|_{\ell_2}}{\alpha}.$$

Furthermore, if $\lambda > 0$ *is a precision level obeying* $\lambda \geq \|\Pi_{\mathcal{S}_+}(\boldsymbol{e})\|_{\ell_\infty}$, *then running gradient descent updates of the from* (2) *with a learning rate* $\eta \leq \frac{1}{2\beta^2} \min\left(1, \frac{\alpha\beta}{L\|\boldsymbol{r}^0\|_{\ell_2}}\right)$, *after* $\tau \geq \frac{5}{\eta\alpha^2} \log\left(\frac{\|\boldsymbol{r}^0\|_{\ell_2}}{\lambda}\right)$ *iterations,* $\boldsymbol{W}^\tau$ *achieves the following error bound with respect to the true labels*

$$\|f(\boldsymbol{W}^\tau, X_S) - \widetilde{\boldsymbol{y}}_S\|_{\ell_\infty} \leq 2\lambda.$$

Finally, if $\boldsymbol{e}$ *has at most* s *nonzeros and* $\mathcal{S}_+$ *is* γ*-diffused i.e., for any vector* $\boldsymbol{v} \in \mathcal{S}_+$ *we have* $\|\boldsymbol{v}\|_{\ell_\infty} \leq \sqrt{\gamma/n}\|\boldsymbol{v}\|_{\ell_2}$ *for some* $\gamma > 0$, *then using* $\lambda = \|\Pi_{\mathcal{S}_+}(\boldsymbol{e})\|_{\ell_\infty}$, *we get*

$$\|f(\boldsymbol{W}^\tau, X_S) - \widetilde{\boldsymbol{y}}_S\|_{\ell_\infty} \leq 2\|\Pi_{\mathcal{S}_+}(\boldsymbol{e})\|_{\ell_\infty} \leq \frac{\gamma\sqrt{s}}{n}\|\boldsymbol{e}\|_{\ell_2}.$$

Lemma A.2 *Let* $\{(\boldsymbol{x}_i, y_i)\}_{i\in S}$ *be the subset selected by* CRUST, *and* $\{\tilde{y}_i\}_{i\in S}$ *be the corresponding noiseless labels. Note that the elements of the selected subsets are mostly clean, but may contain a smaller fraction* ρ *of noisy labels. Let* $\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S)$ *be the Jacobian matrix corresponding to the selected subset which is rank* k, *and* $\mathcal{S}_+$ *be its column space. Then, the difference between noiseless and noisy labels satisfy the bound*

$$\|\Pi_{\mathcal{S}_+}(\boldsymbol{y}_S - \tilde{\boldsymbol{y}}_S)\|_{\ell_\infty} \leq 2\rho.$$

Proof The proof is similar to that of Lemma 8.10 in [23], but using $\mathcal{S}_+$ as defined in Eq. (12). ■

A.1 Proof of Theorem 4.1

We first consider the case where the set $S \subseteq V$ of k weighted medoids found by CRUST approximates the largest singular value of the neural network Jacobian by an error of $\epsilon = 0$. Therefore, we can apply Corollary A.3 to characterize the behavior of gradient descent on the weighted subsets found by CRUST. The following Corollary summarizes the results:

Corollary A.3 *Assume that we apply gradient descent on the least-squares loss in Eq.* (2) *to train a neural network on subsets found by* CRUST *from a dataset with class labels* $\{\nu_j\}_{j=1}^C \in [-1, 1]$, *label margin* δ. *Suppose that the Jacobian mapping is* L*-smooth, and let* $\alpha = \min_\tau \sqrt{r^\tau_{\min}}\sigma_{\min}(\mathcal{J}(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}))$, *and* $\beta = \max_\tau \sqrt{r^\tau_{\max}}\|\mathcal{J}(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau})\|$ *be the minimum and maximum singular values of the Jacobian of the subsets weighted by* $\boldsymbol{r}^\tau$ *found by* CRUST, *during* τ *steps of gradient descent updates. If subsets contain a fraction of* $\rho \leq \delta/8$ *noisy labels, using step size* $\eta = \frac{1}{2\beta^2}\min(1, \frac{\alpha\beta}{L\sqrt{2k}})$, *after* $\tau = \mathcal{O}(\frac{1}{\eta\alpha^2}\log(\frac{\sqrt{k}}{\rho}))$ *iterations, the neural network classifies all the selected elements correctly.*

Proof Fix a vector $\boldsymbol{v}$ and let $\tilde{\boldsymbol{p}} = \mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_S)\boldsymbol{v}$ and $\boldsymbol{p} = \mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S)\boldsymbol{v}$. Entries of $\tilde{\boldsymbol{p}}$ multiply the entries of $\boldsymbol{p}$ somewhere between $r_{\min}$ and $r_{\max}$. This establishes the upper and lower bounds on the singular values of $\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_S)$ over $\mathcal{S}_+$ in terms of the singular values of $\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S)$. I.e.,

$$\sqrt{r_{\min}}\sigma_{\min}(\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_{S^*}), \mathcal{S}_+) \leq \sigma_{i\in[k]}(\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_{S^*}), \mathcal{S}_+) \leq \sqrt{r_{\max}}\|\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_{S^*})\|. \tag{13}$$

Therefore, we get that $\alpha = \min_\tau \sqrt{r^\tau_{\min}}\sigma_{\min}(\mathcal{J}(\boldsymbol{W}^\tau, \boldsymbol{X}^\tau_S)), \beta = \max_\tau \sqrt{r^\tau_{\max}}\|\mathcal{J}(\boldsymbol{W}^\tau, \boldsymbol{X}^\tau_S)\|$.

Moreover, we have $f : \mathbb{R}^d \to [-1, 1]$, and class labels $\{\nu_j\}_{j=1}^C \in [-1, 1]$. Hence, for every element $i \in S$, we have $|f(\boldsymbol{W}, \boldsymbol{x}_i) - y_i| \leq 2$, and the upper-bound on the initial misfit is $\|\boldsymbol{r}^0\|_{l_2} \leq \sqrt{2k}$.

Now, using Lemma A.2, we know that

$$\|\Pi_{\mathcal{S}_+}(\boldsymbol{y} - \tilde{\boldsymbol{y}})\|_{\ell_\infty} \leq 2\rho.$$

Substituting the values corresponding to α, β in Theorem 4.1, we get that after

$$\frac{5}{\eta\alpha^2}\log(\frac{\|\boldsymbol{r}^0\|_{\ell_2}}{2\rho}) \leq \frac{5}{\eta\alpha^2}\log(\frac{\sqrt{2k}}{2\rho}) \leq \tau \tag{14}$$

gradient descent iterations, the error bound with respect to the true labels is $\|f(\boldsymbol{W}_\tau, X_S) - \widetilde{\boldsymbol{y}}_S\|_{\ell_\infty} \leq 2\rho$. If gradient clusters are roughly balanced, i.e., there are $\mathcal{O}(k/K')$ data points in each cluster, we get that for all gradient iterations with

$$\frac{5}{\eta\alpha^2}\log(\frac{\|\boldsymbol{r}^0\|_{\ell_2}}{2\rho}) \leq \frac{5}{\eta\alpha^2}\log(\frac{\sqrt{2k}}{2\rho}) = \mathcal{O}(\frac{K}{\eta k\sigma^2_{\min}\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S)}\log(\frac{\sqrt{k}}{\rho})) \leq \tau, \tag{15}$$

the infinity norm of the residual obeys (using $\lambda = \|\Pi_{\mathcal{S}_+}(\boldsymbol{e})\|_{\ell_\infty} \leq 2\rho$)

$$\|f(\boldsymbol{W}, X_S) - \tilde{\boldsymbol{y}}\|_{\ell_\infty} \leq 4\rho.$$

This implies that if $\rho \leq \delta/8$, the labels predicted by the network are away from the correct labels by less than $\delta/2$, hence the elements of the subsets (including noisy ones) will be classified correctly. ■

A.2 Completing the Proof of Theorem 4.1

Next, we consider the case where weighted subsets found by CRUST approximate the prominent singular value of the Jacobian matrix by an error of at most ϵ. Here, we characterize the behavior of gradient descent by comparing the iterations with and without error.

In particular, starting from $\boldsymbol{W}^0 = \bar{\boldsymbol{W}}^0$ consider the gradient descent iterations on the weighted subsets $\bar{S}$ that estimating the largest singular value of the neural network Jacobian without an error,

$$\bar{\boldsymbol{W}}^{\tau+1} = \bar{\boldsymbol{W}}^\tau - \eta\mathcal{J}_r^T(\bar{\boldsymbol{W}}^\tau, \boldsymbol{X}_{\bar{S}^\tau})(f(\bar{\boldsymbol{W}}^\tau, \boldsymbol{X}) - \boldsymbol{y}_{\bar{S}^\tau}), \tag{16}$$

and gradient descent iterations on the weighted subsets S with an error of at most ϵ in estimating the largest singular value of the neural network Jacobian,

$$\boldsymbol{W}^{\tau+1} = \boldsymbol{W}^\tau - \eta\mathcal{J}_r^T(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau})(f(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}) - \boldsymbol{y}_{S^\tau}). \tag{17}$$

To proceed with the proof, we use the following short hand notations for residuals and Jacobian matrix in iteration τ of gradient descent:

$$\boldsymbol{r}^\tau = f(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}) - \boldsymbol{y}_{S^\tau},\ \bar{\boldsymbol{r}}^\tau = f(\bar{\boldsymbol{W}}^\tau, \boldsymbol{X}_{\bar{S}^\tau}) - \boldsymbol{y}_{\bar{S}^\tau} \tag{18}$$

$$\mathcal{J}^\tau = \mathcal{J}_r(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}),\ \mathcal{J}^{\tau+1,\tau} = \mathcal{J}_r(\boldsymbol{W}^{\tau+1}, \boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}), \tag{19}$$

$$\bar{\mathcal{J}}^\tau = \mathcal{J}_r(\bar{\boldsymbol{W}}^\tau, \boldsymbol{X}_{S^\tau}),\ \bar{\mathcal{J}}^{\tau+1,\tau} = \mathcal{J}_r(\bar{\boldsymbol{W}}^{\tau+1}, \bar{\boldsymbol{W}}^\tau, \boldsymbol{X}_{\bar{S}^\tau}) \tag{20}$$

$$d^\tau = \|\boldsymbol{W}^\tau - \bar{\boldsymbol{W}}^\tau\|_F,\ p^\tau = \|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_F, \tag{21}$$

where $\mathcal{J}_r(\boldsymbol{W}^1, \boldsymbol{W}^2, \boldsymbol{X}_S)$ denotes the average weighted neural network Jacobian at subset $\boldsymbol{X}_S$, i.e.,

$$\mathcal{J}_r(\boldsymbol{W}^1, \boldsymbol{W}^2, \boldsymbol{X}_S) = \int_0^1 \mathcal{J}_r(\alpha\boldsymbol{W}^1 + (1-\alpha)\boldsymbol{W}^2, \boldsymbol{X}_S)d\alpha.$$

We first proof the following Lemma that bounds the normed difference between $\mathcal{J}_r(\bar{\boldsymbol{W}}^1, \bar{\boldsymbol{W}}^2, \boldsymbol{X}_{\bar{S}})$ and $\mathcal{J}_r(\boldsymbol{W}^1, \boldsymbol{W}^2, \boldsymbol{X}_S)$.

Lemma A.4 *Let $\boldsymbol{X}_S, \boldsymbol{X}_{\bar{S}}$ be the subset of data points found by* CRUST*, that approximates the Jacobian matrix on the entire data by and error of* 0 *and ϵ, respectively. Given parameters $\boldsymbol{W}^1, \boldsymbol{W}^2, \bar{\boldsymbol{W}}^1, \bar{\boldsymbol{W}}^2$, we have that*

$$\|\mathcal{J}_r(\boldsymbol{W}^1, \boldsymbol{W}^2, \boldsymbol{X}_S) - \mathcal{J}_r(\bar{\boldsymbol{W}}^1, \bar{\boldsymbol{W}}^2, \boldsymbol{X}_{\bar{S}})\| \leq (\frac{\|\bar{\boldsymbol{W}}^1 - \boldsymbol{W}^1\|_F + \|\bar{\boldsymbol{W}}^2 - \boldsymbol{W}^2\|_F}{2} + \epsilon).$$

Proof Given $\boldsymbol{W}, \bar{\boldsymbol{W}}$, we can write

$$\|\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_S) - \mathcal{J}_r(\bar{\boldsymbol{W}}, \boldsymbol{X}_{\bar{S}})\| \leq \|\mathcal{J}_r(\boldsymbol{W}, \boldsymbol{X}_S) - \mathcal{J}_r(\bar{\boldsymbol{W}}, \boldsymbol{X}_S)\| + \|\mathcal{J}_r(\bar{\boldsymbol{W}}, \boldsymbol{X}_{\bar{S}}) - \mathcal{J}_r(\bar{\boldsymbol{W}}, \boldsymbol{X}_S)\| \tag{22}$$

$$\leq L\|\boldsymbol{W} - \bar{\boldsymbol{W}}\| + \epsilon. \tag{23}$$

To get the result on $\|\mathcal{J}(\boldsymbol{W}^1, \boldsymbol{W}^2, \boldsymbol{X}_S) - \mathcal{J}_r(\bar{\boldsymbol{W}}^1, \bar{\boldsymbol{W}}^2, \boldsymbol{X}_{\bar{S}})\|$, we integrate

$$\|\mathcal{J}(\boldsymbol{W}^1, \boldsymbol{W}^2, \boldsymbol{X}_S) - \mathcal{J}_r(\bar{\boldsymbol{W}}^1, \bar{\boldsymbol{W}}^2, \boldsymbol{X}_{\bar{S}})\| \leq \int^0{}^1(L\|\alpha(\bar{\boldsymbol{W}}^1 - \boldsymbol{W}^1) + (1-\alpha)(\bar{\boldsymbol{W}}^1 - \boldsymbol{W}^1)\|_F + \epsilon)d\alpha \tag{24}$$

$$\leq \frac{L(\|\bar{\boldsymbol{W}}^1 - \boldsymbol{W}^1\|_F + \|\bar{\boldsymbol{W}}^2 - \boldsymbol{W}^2\|_F)}{2} + \epsilon. \tag{25}$$

■

Now, applying Lemma A.4, we have

$$\|\mathcal{J}_r(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}) - \mathcal{J}_r(\bar{\boldsymbol{W}}^\tau, \boldsymbol{X}_{\bar{S}^\tau})\| \le L\|\bar{\boldsymbol{W}}^\tau - \boldsymbol{W}^\tau\|_F + \epsilon \le Ld^\tau + \epsilon \tag{26}$$

$$\|\mathcal{J}_r(\boldsymbol{W}^{\tau+1}, \boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}) - \mathcal{J}_r(\bar{\boldsymbol{W}}^{\tau+1}, \bar{\boldsymbol{W}}^\tau, \boldsymbol{X}_{\bar{S}^\tau})\| \le L(d^\tau + d^{\tau+1})/2 + \epsilon. \tag{27}$$

Following this and since the normed noiseless residual is non-increasing and satisfies $\|\bar{\boldsymbol{r}}_\tau\|_{\ell_2} \le \|\bar{\boldsymbol{r}}^0\|_{\ell_2}$, we can write

$$\boldsymbol{W}^{\tau+1} = \boldsymbol{W}^\tau - \eta\mathcal{J}^\tau\boldsymbol{r}^\tau \quad , \quad \bar{\boldsymbol{W}}^{\tau+1} = \bar{\boldsymbol{W}}^\tau - \eta(\bar{\mathcal{J}}^\tau)^T\bar{\boldsymbol{r}}^\tau \tag{28}$$

$$\|\boldsymbol{W}^{\tau+1} - \bar{\boldsymbol{W}}^{\tau+1}\|_F \le \|\boldsymbol{W}^\tau - \bar{\boldsymbol{W}}^\tau\|_F + \eta\|\mathcal{J}^\tau - \bar{\mathcal{J}}^\tau\|\|\bar{\boldsymbol{r}}^\tau\|_{\ell_2} + \eta\|\mathcal{J}^\tau\|\|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_{\ell_2}, \tag{29}$$

$$d^{\tau+1} \le d^\tau + \eta\big((Ld^\tau + \epsilon)\|\bar{\boldsymbol{r}}^0\|_{\ell_2} + \beta p^\tau\big). \tag{30}$$

For the residual we have

$$\boldsymbol{r}^{\tau+1} = \boldsymbol{r}^\tau - f(\boldsymbol{W}^\tau, \boldsymbol{X}_S) + f(\boldsymbol{W}^{\tau+1}, \boldsymbol{X}_S) \tag{31}$$

$$= \boldsymbol{r}^\tau + \mathcal{J}^{\tau+1,\tau}(\boldsymbol{W}^{\tau+1} - \boldsymbol{W}^\tau) \tag{32}$$

$$= \boldsymbol{r}^\tau - \eta\mathcal{J}^{\tau+1,\tau}(\mathcal{J}^\tau)^T\boldsymbol{r}^\tau, \tag{33}$$

where in Eq. (33) we used $\boldsymbol{W}^{\tau+1} - \boldsymbol{W}^\tau = \eta\nabla\mathcal{L}(\boldsymbol{W}^\tau) = \eta(\mathcal{J}^\tau)^T\boldsymbol{r}^\tau$. Furthermore, we can write

$$\boldsymbol{r}^{\tau+1} - \bar{\boldsymbol{r}}^{\tau+1} = (\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau) - \eta(\mathcal{J}^{\tau+1,\tau} - \bar{\mathcal{J}}^{\tau+1,\tau})(\mathcal{J}^\tau)^T\boldsymbol{r}^\tau \tag{34}$$

$$- \eta\bar{\mathcal{J}}_{\tau+1,\tau}\big((\mathcal{J}^\tau)^T - (\bar{\mathcal{J}}^\tau)^T\big)\boldsymbol{r}^\tau - \eta\bar{\mathcal{J}}^{\tau+1,\tau}(\bar{\mathcal{J}}^\tau)^T(\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau) \tag{35}$$

$$= \big(\boldsymbol{I} - \eta\bar{\mathcal{J}}^{\tau+1,\tau}(\bar{\mathcal{J}}^\tau)^T\big)(\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau) - \eta(\mathcal{J}^{\tau+1,\tau} - \bar{\mathcal{J}}^{\tau+1,\tau})(\mathcal{J}^\tau)^T\boldsymbol{r}^\tau \tag{36}$$

$$- \eta\bar{\mathcal{J}}^{\tau+1,\tau}\big((\mathcal{J}^\tau)^T - (\bar{\mathcal{J}}^\tau)^T\big)\boldsymbol{r}^\tau. \tag{37}$$

Using $\boldsymbol{I} \succeq \bar{\mathcal{J}}^{\tau+1,\tau}(\bar{\mathcal{J}}^\tau)^T/\beta^2 \succeq 0$, we have

$$\|\boldsymbol{r}^{\tau+1} - \bar{\boldsymbol{r}}^{\tau+1}\|_{\ell_2} \le \|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_{\ell_2} + \eta\beta\|\boldsymbol{r}^\tau\|_{\ell_2}(L(3d^\tau + d^{\tau+1})/2 + 2\epsilon) \tag{38}$$

$$\le \|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_{\ell_2} + \eta\beta(\|\bar{\boldsymbol{r}}^0\|_{\ell_2} + p^\tau)(L(3d^\tau + d^{\tau+1})/2 + 2\epsilon), \tag{39}$$

where we used $\|\boldsymbol{r}^\tau\|_{\ell_2} \le p^\tau + \|\bar{\boldsymbol{r}}^0\|_{\ell_2}$ and $\|(\boldsymbol{I} - \eta\bar{\mathcal{J}}^{\tau+1,\tau}(\bar{\mathcal{J}}^\tau)^T)\boldsymbol{v}\|_{\ell_2} \le \|\boldsymbol{v}\|_{\ell_2}$ which follows from the contraction of the residual. This implies that

$$p^{\tau+1} \le p^\tau + \eta\beta(\|\bar{\boldsymbol{r}}^0\|_{\ell_2} + p^\tau)(L(3d^\tau + d^{\tau+1})/2 + 2\epsilon). \tag{40}$$

We use a similar inductive argument as that of Theorem 8.10 in [23]. The claim is that if for all $t \le \tau_0$, we have (using $\|\bar{\boldsymbol{r}}^0\|_{\ell_2} \le \Theta$)

$$\epsilon \le \mathcal{O}\Big(\frac{\alpha^2}{\beta\log(\sqrt{k}/\rho)}\Big) \le \mathcal{O}\Big(\frac{k\sigma^2_{\min}(\mathcal{J}(\boldsymbol{W}, \boldsymbol{X}_S))}{K\beta\log(\sqrt{k}/\rho)}\Big), \quad \text{and} \quad L \le \frac{2}{5\tau_0\eta\Theta(1 + 8\eta\tau_0\beta^2))}, \tag{41}$$

then it follows that

$$p^t \le 8t\eta\epsilon\Theta\beta, \qquad d^t \le 2t\eta\epsilon\Theta(1 + 8\eta\tau_0\beta^2) \le 20t\eta^2\tau_0\epsilon\Theta\beta^2 \le \mathcal{O}(t\eta^2\tau_0 k^{3/2}\epsilon). \tag{42}$$

The proof is by induction. Suppose for $t \le \tau_0 - 1$, we have that

$$p^t \le 8t\eta\epsilon\Theta\beta \le \Theta, \qquad d^t \le 2t\eta\epsilon\Theta(1 + 8\eta\tau_0\beta^2). \tag{43}$$

At $t + 1$, from (30) we know that

$$\frac{d^{t+1} - d^t}{\eta} \le Ld^t\Theta + \epsilon\Theta + 8\tau_0\eta\beta^2\epsilon\Theta \tag{44}$$

Now, using $L \le \frac{2}{5\tau_0\eta\Theta(1+8\eta\tau_0\beta^2))} \le \frac{1}{2\eta\tau_0\Theta}$ from (41), and replacing d^t from (43) into (44) we get

$$\frac{d^{t+1} - d^t}{\eta} \le Ld^t\Theta + \epsilon\Theta + 8\tau_0\eta\beta^2\epsilon\Theta \overset{?}{\le} 2\epsilon\Theta(1 + 8\eta\tau_0\beta^2). \tag{45}$$

This establishes the induction for d^{t+1}.

To show the induction on p^t, following (8.64) and using $p^t \le \Theta$, we need

$$\frac{p^{t+1}-p^t}{\eta} \le \beta\Theta(L(3d^\tau + d^{\tau+1}) + 4\epsilon) \overset{?}{\le} 8\epsilon\Theta\beta \tag{46}$$

$$L(3d^\tau + d^{\tau+1}) + 4\epsilon \overset{?}{\le} 8\epsilon \tag{47}$$

$$L(3d^\tau + d^{\tau+1}) \overset{?}{\le} 4\epsilon \tag{48}$$

$$10L\tau_0\eta(1+8\eta\tau_0\beta^2)\Theta \overset{?}{\le} 4, \tag{49}$$

where in the last inequality we used $3d^t + d^{t+1} \le 10\tau_0\eta\epsilon\Theta(1+8\eta\tau_0\beta^2)$. Note that $\eta = \frac{1}{2\beta^2}\min(1, \frac{\alpha\beta}{L\sqrt{2k}})$. Hence, if $\frac{\alpha\beta}{L\sqrt{2k}} \ge 1$, we get that $\eta = \frac{1}{2\beta^2} \ge \frac{1}{\tau_0\beta^2}$, and thus $\eta\tau_0\beta^2 \ge 1$. This allows upper-bounding $3d^t + d^{t+1}$. Now, for L we have

$$L \le \frac{2}{5\tau_0\eta\Theta(1+8\eta\tau_0\beta^2))}. \tag{50}$$

This concludes the induction since the condition on L is satisfied.

Now, from (44) and using $\eta\tau_0\beta^2 \ge 1$ we get

$$d^t \le 2t\eta\epsilon\Theta(1+8\eta\tau_0\beta^2) \le \mathcal{O}(t\eta^2\tau_0\sqrt{k}\epsilon). \tag{51}$$

Finally, from (43) we have $p^t \le 8t\eta\epsilon\Theta\beta \le \Theta$. Using $\eta\tau_0 = \mathcal{O}(\frac{K}{k\sigma_{\min}^2\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S)}\log(\frac{\sqrt{k}}{\rho}))$ and noting that $\alpha \le \beta$ we have that for $\tau \ge t$

$$\epsilon \le \frac{1}{8\tau_0\eta\beta} \le \mathcal{O}(\frac{\alpha^2}{\beta\log(\sqrt{k}/\rho)}) \le \mathcal{O}(\frac{k\sigma_{\min}^2(\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S))}{K\beta\log(\sqrt{k}/\rho)}).$$

Now we calculate the misclassification error. From Corollary A.3 and Eq. (43) we have that for $\eta = \frac{1}{2\beta^2}\min(1, \frac{\alpha\beta}{L\sqrt{2k}})$ and $\Theta = \sqrt{2k}$, after $\tau = \mathcal{O}(\frac{K}{\eta k\sigma_{\min}^2\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S)}\log(\frac{\sqrt{k}}{\rho}))$ iterations, we get

$$\|\bar{\boldsymbol{r}}^\tau\|_{\ell_\infty} \le 4\rho \quad \text{and} \tag{52}$$

$$\|p^t\|_{\ell_2} = \|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_{\ell_2} \le c\epsilon\frac{K\beta}{\sqrt{k}\sigma_{\min}^2\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S)}\log(\frac{\sqrt{k}}{\rho}). \tag{53}$$

To calculate the classification rate, we denote the residual vectors $\bar{\boldsymbol{r}}^\tau = f(\bar{\boldsymbol{W}}^\tau, \boldsymbol{X}_{\bar{S}^\tau}) - \tilde{\boldsymbol{y}}_{S^\tau}$ and $\boldsymbol{r}^\tau = f(\boldsymbol{W}^\tau, \boldsymbol{X}_{S^\tau}) - \tilde{\boldsymbol{y}}_{S^\tau}$. Now, we count the number of entries of $\boldsymbol{r}^\tau$ that is larger than the label margin $\delta/2$ in absolute value. Let $\mathcal{I}$ be the set of entries satisfying this condition. For $i \in \mathcal{I}$ we have $|r_i^\tau| \ge \delta/2$. Therefore,

$$|\bar{r}_i^\tau| + |r_i^\tau - \bar{r}_i^\tau| \ge |r_i^\tau + \bar{r}_i^\tau - \bar{r}_i^\tau| \ge \delta/2, \tag{54}$$

Since $\rho = (1-\gamma)\delta/8 < \delta/8$ for $0 < \gamma \ll 1$, we get $\|\bar{\boldsymbol{r}}^\tau\|_{\ell_\infty} \le 4\rho \le (1-\gamma)\delta/2$ and

$$|r_i^\tau - \bar{r}_i^\tau| \ge \gamma\delta/2. \tag{55}$$

Thus, the sum of the entries with a larger error than the label margin is

$$\|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_{\ell_1} \ge |\mathcal{I}|\gamma\delta/2. \tag{56}$$

Consequently, we have

$$|\mathcal{I}|\gamma\delta/2 \le \|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_{\ell_1} \le \sqrt{k}\|\boldsymbol{r}^\tau - \bar{\boldsymbol{r}}^\tau\|_{\ell_2} \le c\epsilon\frac{K\beta}{\sigma_{\min}^2\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S)}\log(\frac{\sqrt{k}}{\rho}). \tag{57}$$

Hence, the total number of errors is at most

$$|\mathcal{I}| \le c'\epsilon\frac{K\beta}{\gamma\delta\sigma_{\min}^2\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S)}\log(\frac{\sqrt{k}}{\rho}) = \mathcal{O}(\frac{\epsilon k\beta}{\gamma\delta\alpha^2}\log(\frac{\sqrt{k}}{\rho})). \tag{58}$$

For the network to classify all the data points correctly, we need $|\mathcal{I}| \le c'\epsilon\frac{K\beta}{\gamma\delta\sigma_{\min}^2\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S)}\log(\frac{\sqrt{k}}{\rho}) < 1$. Hence, we get that

$$\epsilon < \frac{c_0\gamma\delta\sigma_{\min}^2(\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S))}{K\beta\log(\frac{\sqrt{k}}{\rho})} < \frac{c_1\delta\sigma_{\min}^2(\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_S))}{K\beta\log(\frac{\sqrt{k}}{\rho})}. \tag{59}$$