We thank all the reviewers for carefully checking the paper and acknowledging the "efficiency and practicality" of our work, and that it "provides theoretical guarantees in an area where recent progress has been largely empirical". Below, we provide clarifications on technical contributions (R1), baseline comparisons under extreme noise (R2, R3), comparison to low-rank approximation (R2), and properties of coresets (R2). We will also clarify in the revised version.

**Technical contribution.** R1 asks for discussion of similarity and difference of technical results to [19]. 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 net. Hence, training on the medoids is robust to noisy labels. In contrast, [19] 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. Crucially, from the clusterable data assumption it easily follows that the neural network covariance $\Sigma(\boldsymbol{X})=\mathbb{E}\big[\big(\phi'(\boldsymbol{X}\boldsymbol{W}^T)\phi'(\boldsymbol{W}^T\boldsymbol{X})\big)\odot\big(\boldsymbol{X}\boldsymbol{X}^T\big)\big]=\frac{1}{k}\mathbb{E}_{\boldsymbol{W}_0}\big[\mathcal{J}(\boldsymbol{W}_0)\mathcal{J}^T(\boldsymbol{W}_0)\big]$ is low-rank, and hence early stopping prevents overfitting noisy labels. While our analysis of the residuals during gradient descent is similar to [19], our results holds for arbitrary deep nets *without early stopping or the above assumptions*.

**More baselines and higher label noise.** R3 asked for comparison to other baselines ([1, 2, 3]), and in extreme cases with even more label noise. The following table compares the average validation accuracy (5 runs) of CRUST compared to L_DMI, T-Revision ([1, 2]) on CIFAR-10 with 20%, 50%, 80% symmetric and 40% asymmetric noisy labels. We see that CRUST outperforms other baselines and shows a significant improvement under severe 80% noise.

| Noise Type | Sym | | | Asym |
|---|---|---|---|---|
| Noise Ratio | 20 | 50 | 80 | 40 |
| L_DMI | $84.3\pm0.4$ | $78.8\pm0.5$ | $36.2\pm1.6$ | $84.8\pm0.7$ |
| T-Revision | $79.3\pm0.5$ | $78.5\pm0.6$ | $20.9\pm2.2$ | $76.3\pm0.8$ |
| CRUST | $\mathbf{91.1\pm0.2}$ | $\mathbf{86.3\pm0.3}$ | $\mathbf{58.3\pm1.8}$ | $\mathbf{88.8\pm0.4}$ |

**Low-rank approximation vs. CRUST.** Minimizing the rank directly as mentioned by R2 ([1, 2, 3]) is impractical for deep nets. Indeed, Eq. 5 finds the best subset of $k$ columns from the Jacobian matrix. However, as discussed in lines 138-145, calculating the Jacobian matrix requires backpropagation on the entire dataset which is very expensive for deep nets. Moreover, finding the best subset of $k$ columns from $\mathcal{J}\in\mathbb{R}^{n\times m}$ has a complexity of poly$(m,n,k)$, where $n, m$ are the number of data points and parameters in the net. Since the subset should be updated at every iteration, the complexity of the above methods becomes prohibitive for deep nets trained on large datasets. Most importantly, while this approach prohibits overfitting noisy labels, it does not help identifying the clean data points. We implemented the greedy column selection of [2] with sketching ($\delta=0.3, \epsilon=0.5$) and lazy evaluation ($\delta=0.5$) on the partial derivatives w.r.t. the last layer (as upper-bounds), but could not finish training CIFAR10 due to the prohibitive running time. We thank R2 for pointing out the related work on low-rank approximation, and will add the discussion to revised version.

**Scalability of CRUST.** R2 asks if CRUSTcan scale to TinyImageNet/full WebvVision/ImageNet. CRUST uses a greedy algorithm to find medoids of each class in the gradient space. The complexity of the greedy algorithm is $\mathcal{O}(nk)$. However, its complexity can be reduced to $\mathcal{O}(n)$ using stochastic methods [25], and can be further improved using lazy evaluation [24] and distributed implementations [27]. 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 the classes, and CRUST can easily scale to large datasets with tens of millions of data points. We will add experiments on larger datasets, such as full WebvVision/ImageNet/Clothing1M suggested by R2 and R4 to the final version.

**Fraction of clean data points in coreset.** R2 asks how clean the coreset is. Fig 1 (a) shows the fraction of clean data points in subsets of size 30%, 50%, and 70% selected by CRUST in presence of 50% noisy labels. It can be seen that CRUST could successfully identify almost 95% of the clean data points, and remove the data corrupted by noisy labels.

**Speedup of training on coresets.** R2 rightly points out the speedup obtained from training on subsets found by CRUST. In fact, CRUST finds the subsets very efficiently, and training on subsets is much faster than on the entire data.

**Squared loss for extreme classification.** R1 asks if squared loss performs well when we have a large number of labels. The key idea of our approach is to find representative data points with clean labels that provide an approximately low-rank Jacobian. If every class has a small number of clean data points, they cluster closely in the gradient space and CRUST will be able to extract clean representative data points. Note that learning from noisy labels would be impossible without assuming a small number of clean data points from every class. At the same time, having a very large number of labels requires selecting larger subsets that contain representative clean data points from all the classes. We note that the use of squared loss is to facilitate the theoretical analysis and our experiments are done using a cross-entropy loss.

**Size of the coreset.** R2 asks if CRUST overfit to noisy labels when $k$ is too large. Ideally, $k$ shouldn't be larger than the number of clean data points. In the limit when we select all the data points, we overfit to noisy labels.

**Notations.** R2 asks for clarification on the following notations: We fix line 160 as $\mathcal{J}_r(\boldsymbol{W},\boldsymbol{X}_{S^*}) = \text{diag}([r_1,\cdots,r_k,0,\cdots 0])\mathcal{J}(\boldsymbol{W},\boldsymbol{X}_{S^*})\in\mathbb{R}^{n\times m}$. $\sigma_i(\mathcal{J}(\boldsymbol{W},\boldsymbol{X}),\mathcal{S}_+)$ is the $i$-th singular value of the Jacobian projected over the support subspace $\mathcal{S}_+$. In Table 2, coreset w/label and w/preds. correspond to finding coresets separately from every class based on their noisy labels, or labels predicted by the model. We thank R2 and will clarify the notations.