Data Pruning: Redundant, Problematic, and Interdependent Samples

Leon Freese[1,2][0009−0001−6973−7831] and Marthinus Wilhelmus Theunissen[1,2,3][0000−0002−7456−7769]

[1] Faculty of Engineering, North-West University, South Africa
[2] Centre for Artificial Intelligence Research, South Africa
[3] National Institute for Theoretical and Computational Sciences, South Africa
{leonfreese5,tiantheunissen}@gmail.com

Abstract. The performance of deep learning models is affected by not only data quantity but also data quality. Data pruning is a process by which practitioners can reduce the size of a dataset by only keeping the most important training data points, thereby achieving similar test set performance. We empirically investigate two popular data pruning methods under noisy and noiseless conditions and show that these methods fail in the presence of significant label noise. We highlight that the success of data pruning is distinctly affected by three factors: redundancy in the dataset, the presence of problematic samples, and interdependence between samples. We perform a detailed investigation on commonly used benchmark classification datasets and neural network architectures. We find that our observations are consistent across data distributions and training protocols.

Keywords: Data pruning · Label noise · Deep learning

1 Introduction

Deep neural networks have garnered significant attention due to their remarkable performance on large-scale datasets. However, their effectiveness fundamentally depends on the quality and quantity of training data [23]. With the growing size and complexity of datasets, it has become increasingly evident that not all samples contribute equally to model performance. Datasets can contain statistical outliers [18], mislabelled samples [12], out of distribution instances [10], redundancies [15], imbalances [17], or even *adversarial* examples [4]. Defining and characterizing how each of these issues affect a model's ability to perform constitutes a wide variety of ongoing investigations in the literature.

Data pruning is the process of removing samples from a train set in a way that allows the model to maintain or improve performance. In addition to possible gains in computational efficiency and robustness, it can also be a useful tool for identifying and analysing training samples to understand their impact on generalization. Data pruning experiments typically involve three steps: 1) ranking training samples based on a *score*, 2) removing a subset of increasing

size based on said ranking, and 3) retraining the model on the reduced dataset to evaluate its performance on a test set. This approach has proven effective in variety of investigations. See Section 2 for related work.

In this work we contribute the following:

- A thorough investigation of the typical data pruning approach.
- We show that the order in which samples should be pruned is not as clear as often motivated in the literature since what constitutes "important" samples is dependent on confounding factors like redundancy and contradictory samples.
- We highlight the large degree of influence the presence of some samples might have on the contribution (to model performance) of others.

In addition to Section 2 summarizing works related to data pruning, Section 3 describes our experimental methods in detail. In Section 4 we present our experimental results along with discussions of our findings. In the final section we conclude with a summarization of our findings.

2 Related work

Most data pruning experiments are done in the context of **data valuation**; that is the task of assigning value to specific samples in the dataset. This value[4] usually refers to how much each sample influences or contributes to the model's overall level of performance [15, 20, 22, 11]. These *scores* might be useful for determining how much individuals should be compensated for their data [3], but they can also be useful for domain adaptation, corrupted sample discovery, and robust learning [22].

Ghorbani and Zou [3] develop a scoring method based on Shapley values. Using a basic data pruning experiment they show that, if samples are pruned in descending order, there is a more pronounced drop in performance than if pruned in a random order or based on a classic baseline scoring method. They also compare performance if samples are pruned in ascending order. In this case the performance tends to improve with each increment. This indicates that their method scores samples in a way that correlates with their contribution to overall model performance. This approach was later generalized by Kwon et al. [9] to similar effect.

Yoon et al. [22] propose an alternative scoring method using a reinforcement learning approach. Using a similar data pruning experiment to Ghorbani and Zou, they show that their scoring method outperforms that of Ghorbani and Zou and other baselines on common benchmarks and label corrupted alternatives.

Paul et al. [15] propose two scoring methods based on gradient- and error dynamics measured early in training. Their data pruning experiments show that when samples are pruned in ascending order, the model maintains good performance at much lower train set sizes than when samples are pruned randomly.

[4] This is often referred to as a *score*; a convention we will be continuing in this work.

Interestingly, they find that label noise tends to increase their scores, but don't show the effect on an actual pruning experiment. More recently, Ki et al. [11] proposed a training-free alternative that performs competitively without having to train a model in order to generate scores.

Data pruning experiments have also been used to evaluate **label error detectors** specifically. For example, Pleiss et al. [16] propose a method for detecting problematic (i.e. mislabelled) samples using an *Area Under the Margin (AUM)* statistic. They use a basic data pruning experiment to show that, when samples are ranked according to this statistic, the model performance improves above an alternative random ranking.

Northcutt et al. [13] propose a method of detecting label errors based on the class output probabilities of a trained model. They evaluate this approach through a variation of data pruning experiments. Their results show that pruning progressively larger portions of the samples detected as label errors improves model performance more than pruning random samples.

Other works do not propose data valuation approaches or label error detectors directly, but use data pruning experiments to either probe the inner workings of machine learning models or use it as a tool to analyse expected model performance under different conditions.

Toneva et al. [21] explore the nature of sample forgetting in deep neural networks. By means of data pruning experiments, they show that if samples are pruned in ascending order of the number of forgetting events, the model is able to maintain full-set performance at smaller train set sizes than if the samples are pruned randomly. This suggests that samples with more forgetting events are more important for model performance. Toneva et al. also mention that label noise tends to increase their scores without demonstrating the implications on pruning experiments.

Looking into how deep neural networks treat individual samples during training, Jiang et al. [7] propose a theoretical score that quantifies the expected accuracy on a held-out sample when train sets of varying sizes are drawn from the data distribution. They show that an approximated version of their score can be used as a successful scoring method for data pruning when compared to a random baseline.

Recently, Sorscher et al. [20] argue that with powerful data pruning metrics we are able to break beyond the power law scaling [5] that is often observed when comparing performance while varying model- or dataset size. They support this argument through an extensive set of data pruning experiments, which include ten different pruning approaches. Their results show that most methods fail to scale to large datasets such as ImageNet [1]. Moreover, the most effective approaches are often computationally expensive. In addition, they develop a theoretical framework to characterize data pruning under simplified conditions (perceptron learning in the student–teacher setting). This framework suggests that when the initial dataset is large, one should prune "easy" samples first, whereas when the dataset is small, one should prune "hard" samples first. Here

easy would correspond to large margins in the teacher and hard samples would correspond to small margins.

We note that none of the works mentioned above pruned the train set to near 100% while comparing with a baseline. We also note that few of the works perform data pruning experiments with explicit label noise. The exceptions being Yoon et al [22], and the two label error detectors Pleiss et al [16]. and Northcutt et al. [13]. Finally, we note the works by Paul et al. [15], Ki et al. [11], and Toneva et al. [21], all mention that their scores tend to be higher for label noise. However, they maintain the argument that samples with higher scores tend to be more important for model performance (hence why they prune in ascending order of score).

These points prompted us to take a closer look at how two popular scoring methods [15, 21] perform as data pruning approaches with and without label noise across the full range of the train set that can be pruned.

3 Methods

Problem formulation In a multiclass classification problem, let the input be $\mathbf{x} \in X \subseteq \mathbb{R}^d$ and the label space $Y = \{1, 2, \ldots, c\}$. The labelled dataset is defined as $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each pair $(\mathbf{x}_i, y_i)$ is called a *sample*. D is divided into two non-overlapping subsets: $D_{\text{train}} \cup D_{\text{test}}$. We assume that D_{test} is entirely drawn *independent and identically distributed (i.i.d.)* from some target distribution over $X \times Y$. We do not make the same assumption for D_{train} due to potential noise.

Data pruning As is convention in the literature, we use a framework containing a model architecture $\mathcal{M}$, a training process $\mathcal{T}$, a scoring method $\mathcal{S}$, and a performance measure $\mathcal{P}$. For D_{train}, the training process is defined as $\mathcal{T}(D_{\text{train}}, \mathcal{M}) \rightarrow \mathcal{M}_{\text{train}}$ where $\mathcal{M}_{\text{train}}$ is the model trained on D_{train}. Similarly, for D_{train}, the performance metric is defined as $\mathcal{P}(D_{\text{train}}, \mathcal{M}_{\text{train}}) \rightarrow p_{\text{train}}$ and represents the overall model performance on the given dataset. The scoring method is defined as $\mathcal{S}(\mathcal{T}) \rightarrow \{s_i\}_{i=1}^n$ where each $s_i \in \mathbb{R}$ is the score assigned to the training sample $(\mathbf{x}_i, y_i) \in D_{\text{train}}$. The data pruning evaluation framework outlined in Algorithm 1 produces a ψ-curve that represents the expected model performance, on a held-out test set, as an increasing portion of the train set is pruned.

If $\mathcal{S}$ produces scores that correlate with how much each training sample contributes to the overall model performance, the ψ-curve should either maintain stable performance or have an initial upwards trend as more samples are being pruned. This is because Algorithm 1 ranks the train samples in ascending order, meaning the least likely to contribute to model performance should be pruned first. In general, an unbiased ranking of the train set should correspond to a gradual downward slope in the ψ-curve since model performance is known to be dependent on the train set size.

By using different approaches to scoring and ranking the train set we can compare data pruning metrics and investigate the role of specific samples in the train set and how they contribute to model performance.

Algorithm 1: Generating a Ψ-curve

Input: D_{train}, D_{test}, $\mathcal{M}$, $\mathcal{T}$, $\mathcal{S}$, and $\mathcal{P}$ as defined above. As well as r, a resolution term for which $1 \leq r \leq |D_{\text{train}}|$ and $|D_{\text{train}}|$ mod $r = 0$.

Output: A ψ-curve of length $|D_{\text{train}}|/r$.

$\mathbb{Z} = argsort(\mathcal{S}(\mathcal{T}(D_{\text{train}}, \mathcal{M})))$

$\psi \leftarrow ()$

for i *in* $(0, ..., |D_{train}|/r)$ **do**

  $D_{\text{trunc}} = D_{\text{train}} - D_{\text{train}}^{\mathbb{Z}^{\{0,\ldots,r\times i\}}}$

  $\mathcal{M}_{trunc} = \mathcal{T}(D_{\text{trunc}}, \mathcal{M})$

  $p_{\text{test}} \leftarrow \mathcal{P}(D_{\text{test}}, \mathcal{M}_{\text{trunc}})$

  $\psi \leftarrow \psi + p_{test}$

return ψ

3.1 Scoring methods

In this work we make use of two established scoring methods from the literature: the *sample forgetting score* of Toneva et al. [21] and the *error L_2-norm (EL2N) score* from Paul et al. [15]. Both of these methods are well-cited and are often compared with by other works [7, 11, 20]. Additionally, they have been shown to perform at a similar level to alternatives at scale in the extensive comparison by Sorscher et al. [20], even producing similar rankings to some alternatives.

Sample Forgetting Score Toneva et al. [21] propose a scoring method based on the forgetting behaviour of neural networks. The method is inspired by catastrophic forgetting, where a model loses previously learned information when trained on new tasks. In the current context, each iteration in the stochastic gradient descent process is treated as a *mini-task*. A training sample is considered *forgotten* at iteration t if the model correctly classifies it at iteration $t-1$, but misclassifies it at t. Here, iterations correspond to consecutive mini-batches containing the sample. To simplify computation, Toneva et al. [21] use epochs as a lower bound for iterations. The forgetting events for each sample are summed across the full training cycle to compute a cumulative *forgetting score* s_i. Samples that are learned once and never forgotten are termed *unforgettable* and receive a score of zero, while samples that are never correctly classified receive an *infinite* forgetting score.

In other words, a sample $(\mathbf{x}_i, y_i) \in D_{\text{train}}$ observed over e iterations during the training process $\mathcal{T}$ is given a *forgetting score* of:

$$s_i = \sum_{t=1}^{e} \mathbf{1}_{\{f_t(\mathbf{x}_i) \neq y_i \& f_{t-1}(\mathbf{x}_i) = y_i\}} \tag{1}$$

where $f_t(\mathbf{x}_i)$ is the model's class prediction for input $\mathbf{x}_i$ at iteration t.

Error L_2-Norm (EL2N) Score Paul et al. [15] introduce a score, which leverages early-training model behaviour. This score is simply the average Euclidean norm of the error vector of the model's prediction with respect to the current sample at a relatively early stage in training. More specifically, for a training sample $(\mathbf{x}_i, y_i) \in D_{\text{train}}$, the *EL2N score* is defined as:

$$s_i = \mathbb{E}\|\mathbf{f}_t(\mathbf{x}_i) - \mathbf{y}_i\|_2 \tag{2}$$

where $\mathbf{f}_t(\mathbf{x}_i)$ denote the model's output probability vector at iteration t for input $\mathbf{x}_i$ and $\mathbf{y}_i$ is a one-hot-encoding of y_i. The expected value is estimated by averaging across multiple model initializations.

Note that both the *sample forgetting score* and the *EL2N score* are grounded in intuitions about the ease with which a sample is fitted by the model. The former is based on how many times a model unlearns the features required to correctly classify it and the latter estimates how wrong the model is on producing the correct output value when training iterations are limited. This aligns with the views of Sorscher et al. [20] about the best strategy for pruning being to prune easy samples first when the data is abundant. Both Toneva et al. [21] and Paul et al. [15] prune in ascending order of their scores.

3.2 Experimental setup

Datasets We conduct experiments on three multiclass classification datasets: a synthetic dataset and two benchmark datasets, MNIST [2] and CIFAR-10 [8]. For the synthetic dataset, we generate samples $(\mathbf{x}_i, y_i)$ such that $\mathbf{x}_i \in \mathbb{R}^{100}$ is uniformly sampled from one of ten class-specific isotropic Gaussian distributions $\{\mathcal{N}(\boldsymbol{\mu}_j, \sigma_j^2\mathbf{I})\}_{j=1}^{10}$, where $\sigma_j \sim \mathcal{U}(1, 5)$, and all $\boldsymbol{\mu}_j$ are separated by a Euclidean distance of seven. Our train- and test set sizes are reported in Table 1. The MNIST and CIFAR-10 train sets are randomly sampled from the full sets and the test sets are the predefined evaluation datasets. For experiments involving label noise, we apply it only to the train set. The noise is applied symmetrically by randomly replacing the labels of a subset of the training samples with other labels randomly chosen from the label space.

Models All models are trained using the Adam [14] optimizer with a batch size and learning rate as specified in Table 1. Batch normalization [6] layers are applied after each hidden layer, before a ReLU activation function. For the

synthetic and MNIST datasets we train Multilayer Perceptron (MLP) models, and for CIFAR-10 we train VGG-16 models. The MLP models consists of two hidden layers, each with 512 nodes. For the VGG-16 models, the last 3 layers of the model are three fully connected layers with $4,096$, $4,096$ and $1,000$ nodes, as suggested by Simonyan et al. [19].

Scoring We compute the scores $\{s_i\}_{i=1}^n$ using each of the two scoring methods by averaging the results over ten randomly initialized models. To count the forgetting events as in Eq 1, each model is trained for 200 epochs. The error vectors for Eq 2 are measured after 20 epochs of training. These design choices were made to closely resemble the setups in the works by Toneva et al. [21] and Paul et al. [15].

Pruning After calculating a score for each training sample, we use Algorithm 1 to generate a ψ-curve. In practice, at each point along the curve, we evaluate the performance over three random initializations of the model architecture. Importantly, we use the same model architecture $\mathcal{M}$ and training process $\mathcal{T}$ as the ones used to calculate the scores. The resolution parameter r is set as shown in Table 1. In addition to the two scoring methods, we perform data pruning experiments using a random baseline, where the sample order ($\mathbb{Z}$ in Algorithm 1) is replaced with a random permutation.

4 Results

From Fig. 1 we note that we can prune roughly 40%, 90%, and 40% of the synthetic, MNIST, and CIFAR-10 train set, respectively, with negligible reductions in the expected model performance. This aligns with others' work where it is found that performance can be maintained while pruning 80% [21] of the full MNIST train set and 30% [21] to 50% [15] of the full CIFAR-10 train set. There is clearly a high level of redundancy in these datasets.

Here we point out that when considering the full range of the horizontal axes in Fig 1, the improvements of the two pruning approaches (blue and orange) over the random baseline (gray) appear less significant than when only focussing on the left side of the plots. At the previously mentioned thresholds, we see a gain of approximately 2% accuracy for CIFAR-10 and even less for MNIST and the synthetic dataset.

Additionally, we see that the random baseline maintains higher model performance at extremely low train set sizes. If the general intuition is that the samples pruned last are *important* for generalization, why would the random baseline models outperform the two scoring methods so consistently in the most extreme cases (i.e. when only keeping the most important samples)? We hypothesize that the reason for this observation is that redundancy can be pruned, but not removed entirely. Samples that might be redundant at large train sizes can still be important for generalization at small train sizes. The random baseline

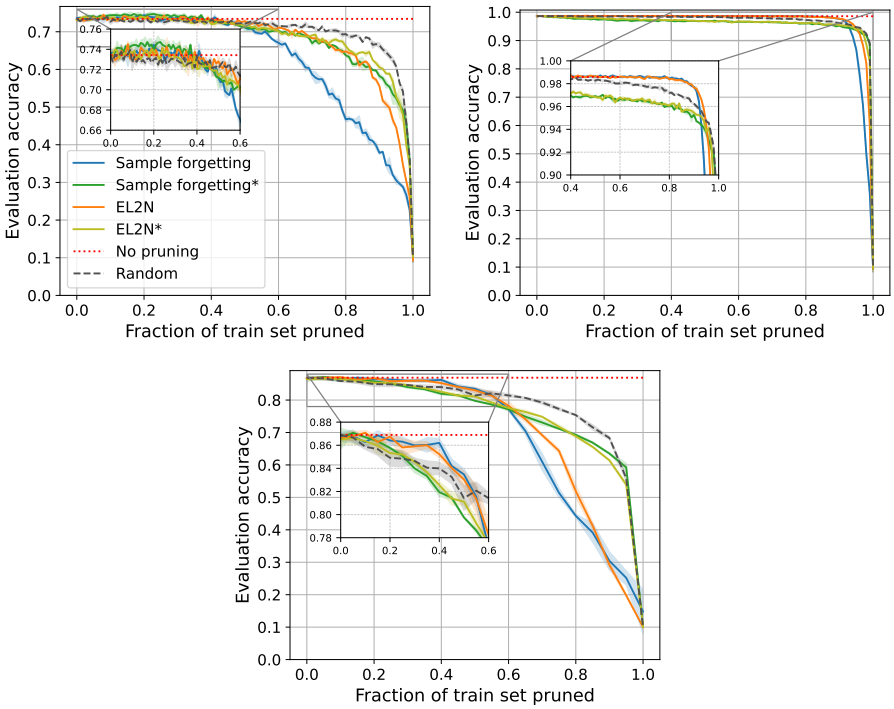

Fig. 1: Model performance as a function of pruning an increasing portion of the train set for synthetic (left-top), MNIST (right-top), and CIFAR-10 (bottom) datasets. Samples are pruned in ascending order, with rankings provided by the two scoring methods in Section 3.1, or by a random baseline. The methods with a $*$ in the title have their ranking reversed. The legend in the first plot is applicable to all three plots.

prunes samples arbitrarily, resulting in samples that were redundant at large train sizes still being present at small train sizes. In contrast, the scoring methods excessively prune redundant samples at large train sizes.

To support this hypothesis we reverse the order of the rankings of the two scoring methods and repeat the pruning experiments. This is indicated by the $*$'d (green and olive) curves. In this case, when an extreme portion of the train set is pruned (only keeping the "least important" samples), the models perform better than when keeping the "most important" samples. This contradiction somewhat aligns with the framework of Sorscher et al. [20] stating that one should prune easy samples when data is abundant and prune hard samples when data is scarce. However, the fact that the models pruned by the random baseline outperforms all the scoring methods in the most extreme cases of data pruning suggests that one requires a mixture of easy and hard samples when pruning to such an extent.

The main takeaway is that when the train set contains substantial redundancy, many samples can be removed (even with random pruning) with little impact on model performance. However, excessive pruning using more powerful scoring methods can cause disproportionate performance drops, as redundant samples are not necessarily inherently unimportant.

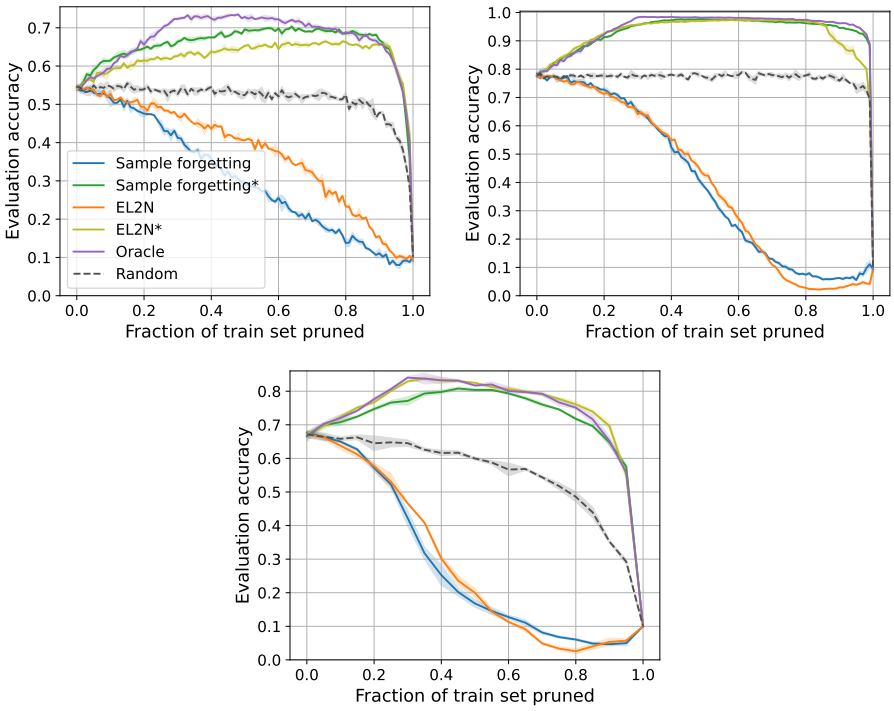

Fig. 2: Model performance as a function of pruning an increasing portion of the train set with 30% label noise for synthetic (left-top), MNIST (right-top), and CIFAR-10 (bottom) datasets. Samples are pruned in ascending order, with rankings provided by the two scoring methods in Section 3.1. The methods with a $*$ in the title have their ranking reversed. The *Random* method provides a random baseline. The *Oracle* method first prunes label corrupted samples in a random order, and then prunes the rest of the train set in a random order. The legend in the first plot is applicable to all three plots.

In Fig. 2 we see that when using the original *sample forgetting* (blue) and *EL2N* (orange) scoring methods in the presence of strong label noise, model performance degrades catastrophically during pruning. In the work by Tenova et al. [21] it is mentioned that label corrupted samples tend to have higher forgetting scores. In the work by Paul et al. [15] it is mentioned that training with

only high scoring samples in a noisy train set might not be optimal. Building on their observations, we analyse the ranking of samples using *EL2N* and *Sample forgetting* scores for our clean and noisy samples before and after label are corrupted in Fig. 3 in Appendix A. We find that label corrupted samples have a strong tendency to be ranked last (therefore dropped last). We also see that there is a tendency for samples that are never corrupted to maintain their relative position in the ranking after the dataset is corrupted. In contrast corrupted samples that were ranked last before corruption tend to be ranked earlier after corruption. The clarity of these tendencies vary across the three datasets, but it can be noticed consistently.

By considering these observations we find that a simple trick to overcome the drop in performance is to reverse the ranking order before pruning the dataset. This is represented in Fig. 2 with the green and olive curves. Note that when the train set is pruned in descending order we not only see that the model performance is maintained, but improves initially as the label noise is ranked higher, therefore being pruned first. Strikingly, for MNIST and CIFAR-10, the *EL2N score* produces model performances similar to an Oracle ranking which first prunes label noise before pruning clean samples.

5 Conclusion

In this work we have performed a controlled set of data pruning experiments to investigate a neglected aspect of data pruning: the influence of label noise on the actual ranking of scoring methods. In the setting that we perform our investigation, we found that:

1. When the data set contains a lot of redundancy, the use of *sample forgetting* and *EL2N* scoring methods produces marginal gains over a random baseline under moderate pruning.
2. At extreme levels of data pruning, the random baseline produces models that outperform those pruned with the *sample forgetting* and *EL2N* scoring methods. This is notable because these methods are specifically designed to retain the most important samples for model performance.
3. Reversing the ranking from *sample forgetting* and *EL2N* scores has little effect under moderate pruning, but paradoxically improves model performance under extreme pruning. This indicates that a sample's score depends on the presence of other samples in the dataset.
4. In the presence of label noise the *sample forgetting* and *EL2N* scoring methods fail completely, but simply reversing the ranking orders results in drastic improvements to model performance.

Our study has certain limitations.

– Whether our findings generalize to larger datasets (e.g. ImageNet) is undetermined.

– Our use of artificial label noise might produce an overly optimistic characterization of the expected performance of these pruning experiments. In real-world scenarios, a broader range of problematic samples exists, each with the potential to introduce unforeseen effects on pruning experiments.
– Our experiments are also all performed on image classification or image classification adjacent problems. Whether the findings extend to other domains, such as time-series regression, remains uncertain.

We hope that these observations guide future data pruning metric development as motivated by Sorscher et al. [20] and others. Future work includes incorporating knowledge of sample interdependence into the ranking process, exploring alternative scoring methods and their sensitivity to label noise, and examining the effects of other types of problematic samples.

Acknowledgment This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Reference Numbers: RCDL240215206999).

A Appendix

Table 1: Dataset sizes and model configurations for the three datasets.

Dataset	$\lvert D_{\text{train}}\rvert$	$\lvert D_{\text{test}}\rvert$	r	Model	Batch size	Learning rate
Synthetic	10,000	5,000	100	2-layer MLP	100	0.01
MNIST	55,000	10,000	550	2-layer MLP	64	0.01
CIFAR-10	45,000	10,000	2,250	VGG-16	64	0.001

References

1. Deng, J., et al.: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009)
2. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**, 141–142 (2012)
3. Ghorbani, A., Zou, J.: Data shapley: Equitable valuation of data for machine learning. In: International Conference on Machine Learning. pp. 2242–2251 (2019)
4. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
5. Hoffmann, J., et al.: Training compute-optimal large language models (2022), preprint at https://arxiv.org/abs/2203.15556
6. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)

7. Jiang, Z., et al.: Characterizing structural regularities of labeled data in overparameterized models. In: International Conference on Machine Learning. pp. 5034–5044 (2021)
8. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009), master's thesis
9. Kwon, Y., Zou, J.: Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In: International Conference on Artificial Intelligence and Statistics. pp. 8780–8802 (2022)
10. Lee, K., et al.: A simple unified framework for detecting out-of-distribution samples and adversarial attacks. Advances in Neural Information Processing Systems **31** (2018)
11. Nohyun, K., Choi, H.and Chung, H.W.: Data valuation without training of a model. In: International Conference on Learning Representations (2022)
12. Northcutt, C.G., Athalye, A., Mueller, J.: Pervasive label errors in test sets destabilize machine learning benchmarks. In: Proceedings of the 35th Conference on Neural Information Processing Systems Track on Datasets and Benchmarks (2021)
13. Northcutt, C.G., Jiang, L., Chuang, I.L.: Confident learning: Estimating uncertainty in dataset labels. Journal of Artificial Intelligence Research **70**, 1373–1411 (2021)
14. P., D., et al.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)
15. Paul, M., et al.: Deep learning on a data diet: Finding important examples early in training. Advances in Neural Information Processing Systems **34**, 20596–20607 (2021)
16. Pleiss, G., et al.: Identifying mislabeled data using the area under the margin ranking. Advances in Neural Information Processing Systems **33**, 17044–17056 (2020)
17. Ren, M., et al.: Learning to reweight examples for robust deep learning. In: International Conference on Machine Learning. pp. 4334–4343 (2018)
18. Sehwag, V., Chiang, M., Mittal, P.: Ssd: A unified framework for self-supervised outlier detection. In: International Conference on Learning Representations (2021)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations. pp. 1–14 (2015)
20. Sorscher, B., et al.: Beyond neural scaling laws: beating power law scaling via data pruning. Advances in Neural Information Processing Systems **35**, 19523–19536 (2022)
21. Toneva, M., et al.: An empirical study of example forgetting during deep neural network learning. In: International Conference on Learning Representations. pp. 1768–1785 (2019)
22. Yoon, J., Arik, S., Pfister, T.: Data valuation using reinforcement learning. In: International Conference on Machine Learning. pp. 10842–10851 (2020)
23. Zhang, C., et al.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations. pp. 3001–3015 (2017)

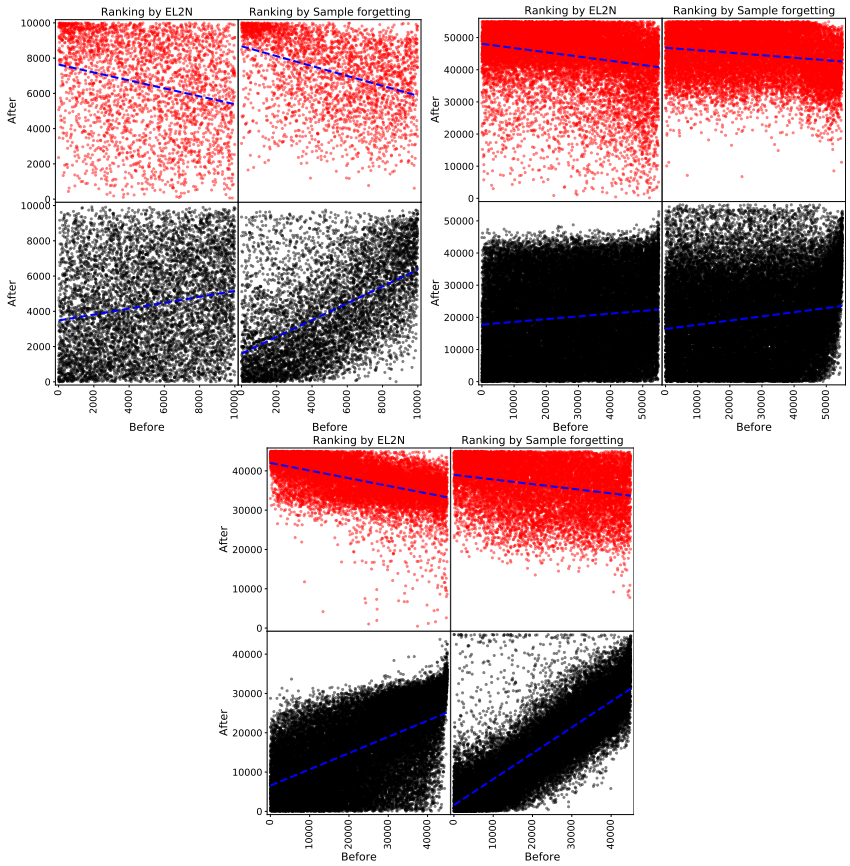

Fig. 3: Sample ranking for synthetic (top-left), MNIST (top-right), and CIFAR-10 (bottom) before and after label noise is applied. Red samples have been label corrupted. Black samples were not corrupted. The blue dashed line is a linear best-fit line. The first column, in each plot, considers error L_2-norm (EL2N) scores. The second column considers sample forgetting scores.