PRIORITIZED TRAINING ON POINTS THAT ARE LEARNABLE, WORTH LEARNING, AND NOT YET LEARNED

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce *reducible held-out loss selection* (RHOLS), a technique for faster model training which selects a sequence of training points that are "just right". We propose a tractable information-theoretic selection function—the reducible held out loss—to efficiently choose training points that maximize information about a holdout set. We show that the "hard" (e.g. high loss) points usually selected in the optimization literature are typically noisy, leading to deterioration on real-world, noisy datasets. At the same time, "easy" (e.g. low noise) samples often prioritized for curriculum learning confer less information. In contrast, RHOLS chooses points that are "just right" and trains in fewer steps than the above approaches.

1 INTRODUCTION

State-of-the-art models such as GPT-3 (Brown et al., 2020), CLIP (Radford et al., 2021), and ViT (Dosovitskiy et al., 2021) achieve remarkable results by training on vast amounts of web scraped data. Especially for large models, training on such datasets requires excessive training times that are measured in weeks or months rather than hours (Radford et al., 2021; He et al., 2020). Even practitioners who develop modestly sized models face long waiting times, due to numerous iterations needed for hyperparameter and architecture choice. These considerations highlight that the total wall clock time required for training is a core constraint in the development of deep learning models.

Accordingly, practitioners with sufficient resources would use many more machines if it further sped up training. However, there are rapidly diminishing returns to using larger batches and distributing them across more workers (Anil et al., 2018; LeCun et al., 2012; Keskar et al., 2016).

At the same time, practitioners have long realized that not all samples in real-world datasets are equally useful (Figure 1, left). Many samples are *noisy*, i.e. mislabelled or inherently ambiguous. For example, the text associated with a web scraped image is rarely an accurate description. Other samples are *redundant*, for example due to the typical overrepresentation of certain object classes in web scraped data (Tian et al., 2021). Such redundant samples can often be left out without losing performance. In many cases, data is so abundant that state-of-the-art performance is reached in less than a single epoch (Brown et al., 2020; Kaplan et al., 2020). With such abundance, it can easily be afforded to skip less useful points.

There are different approaches to selecting the most useful datapoints for training. Some bodies of literature, including curriculum learning, suggest prioritizing *easy* points with low noise before training on all points equally (Bengio et al., 2009). While this can improve convergence and generalization, it fails to deprioritize redundant points. Other works instead suggest to train on points that are particularly *hard* for the current model. In the optimization literature, online batch selection methods (Loshchilov & Hutter, 2015) do this by selecting points with a high loss or gradient norm. Such online selection can successfully skip redundant points.

However, we show that prioritising hard examples can be detrimental on noisy data. In real-world datasets, examples may induce a high loss because they are mislabelled or inherently ambiguous. Indeed, in controlled experiments, with just 10% uniform label noise, "hard" points are overwhelmingly those where the label is corrupted (Figure 1b). Our experiments show that prioritising them degrades performance severely. Additionally, some samples are difficult because they are outliers with unusual features, far from the other training samples. Such points can also lie far from the test data, which makes them less *relevant* for reducing the test loss.

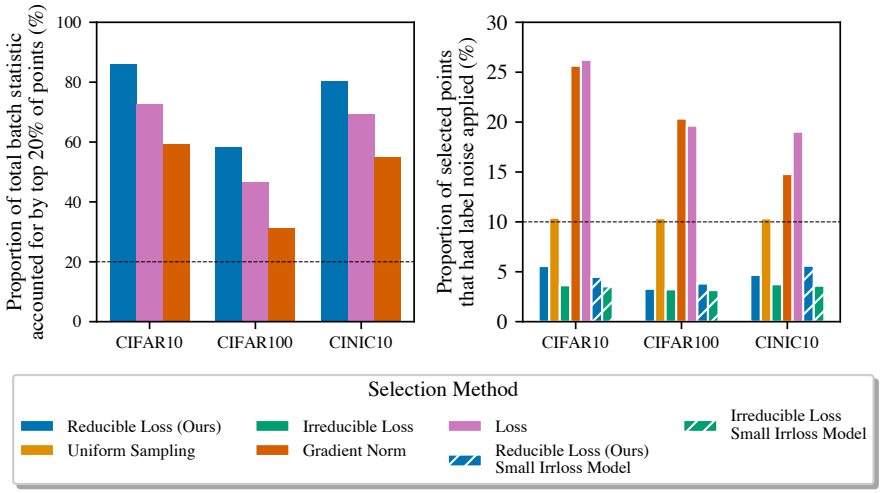

Figure 1: Properties of different selection functions: reducible loss (ours), irreducible loss, loss, and gradient norm. Results are averaged over 15 training epochs. **Left:** The top 20% of points in each batch are responsible for most of the total (reducible) loss and gradient norm summed across the batch, suggesting using such metrics to select training points. **Right:** Proportion of selected points with realized label noise. We added 10% uniform label noise, i.e., we randomly switched each point's label with 10% probability. Selecting the 10% of points with highest loss or gradient norm gives batches where $> 10\%$ of the labels were noisy. This contrasts with reducible and irreducible loss, even when approximating the irreducible loss with a model 21x smaller than the target model to train.

To overcome these limitations, we introduce *reducible held-out loss selection* (RHOLS). Building on recent progress in active learning (Kirsch et al., 2021), we introduce a selection function—Predictive Information Content (PIC)—that quantifies the information each point confers about the labels of a heldout set. We derive an intuitive, easy-to-implement novel selection function: reducible held-out (cross-entropy) loss. By maximizing the information gained about the labels of a held-out set, our method chooses non-redundant, relevant points with low noise—points that are "just right".

We explore the effects of reducible loss selection on four datasets, with and without additional label noise. We evaluate the reduction in required training steps as compared to conventional uniform sampling and a number of recent selection methods. Our method trains in 1.76x-3.82x fewer steps than conventional training on curated benchmarks. In the presence of just 10% label noise, this speedup increases further to 2.34x-5.15x, while other methods degrade. We make no claim that our method improves efficiency in all practical settings or at dataset sizes we have not explored. Instead, we view these results as explaining why prioritizing "hard" samples has not caught on with real-world data, and as a proof of concept that large speedups are still possible.

2 Background: Online batch selection

Consider a model $\mathrm{p}(y \mid x, \theta)$ trained on data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ using stochastic gradient descent. At each training step t, we load a batch b_t of size n_b from $\mathcal{D}$. The labels $\{y_i\}_{i=1}^n$ may be given, as in supervised learning, or could be self-supervised, e.g. the sequence of next words in a document or the rotation an image patch has undergone. In online batch selection (Loshchilov & Hutter, 2015), we pre-sample a larger batch B_t of size $n_B > n_b$ and rank the points x_i in this batch according to a label-aware selection function $S(x_i, y_i)$. Then, we construct a smaller batch b_t that consists of the top-ranking n_b points in B_t and perform one gradient step to minimize a mini-batch loss $\sum_{i \in b_t} L(y_i, \mathrm{p}(y_i \mid x_i, \theta))$. The following large batch B_{t+1} is then pre-sampled from $\mathcal{D}$ without replacement of the previously sampled points (points are replaced only at the start of the next epoch).

We work with this approach for its simplicity, convergence guarantees, and strong performance in recent work (Kawaguchi & Lu, 2020). However, it can be extended to incorporate stochastic data selection (Jiang et al., 2019) with importance weights (Schaul et al., 2015; Katharopoulos & Fleuret,

2018) or to include earlier points x_i in B_t whose selection score $S(x_i, y_i)$ has been previously computed (Loshchilov & Hutter, 2015). Without importance weights, the simple approach above biases the location of the minimum of the loss. However, this biased selection can improve test performance both in theory and practice (Farquhar et al., 2021; Kawaguchi & Lu, 2020).

Notation. For a holdout set $\{y_i^{\text{ho}}\}_i, \{x_i^{\text{ho}}\}_i$ we will write $\mathbf{y}^{\text{ho}}, \mathbf{x}^{\text{ho}}$ etc. for brevity. Note that lower-case variables like y are outcomes and upper-case variables like Y are random variables.

3 Reducible held-out loss—a measure of predictive information content

Online batch selection methods construct batches at training time using points that have high loss or high gradient norm (Loshchilov & Hutter, 2015; Katharopoulos & Fleuret, 2018). In this section, we derive and discuss an alternative selection function, based on an information-theoretic perspective.

Predictive Information Content (PIC). Our overall objective is to select a point $(x, y) \in \mathcal{D}$ that minimizes the expected loss $\mathbb{E}_{\mathrm{p}_{\text{true}}(x,y)}\, \mathrm{h}[y \mid x, \mathcal{D}_{\text{t}}]$ of the model, where

$$\mathrm{h}[y \mid x, \mathcal{D}_{\text{t}}] := -\log \mathrm{p}(y \mid x, \mathcal{D}_t) \tag{1}$$

denotes the cross-entropy loss on (x, y) of a model trained on $\mathcal{D}_{\text{t}}$. As a proxy, we aim to maximize the reduction in the Monte Carlo expectation of the cross-entropy loss on a held-out dataset $\mathcal{D}_{\text{ho}}$,

$$\frac{1}{|\mathcal{D}_{\text{ho}}|} \sum_{i \in \mathcal{D}_{\text{ho}}} \mathrm{h}[y_i^{\text{ho}} \mid x_i^{\text{ho}}, \mathcal{D}_{\text{t}}] - \mathrm{h}[y_i^{\text{ho}} \mid x_i^{\text{ho}}, \mathcal{D}_{\text{t}} \cup (x, y)]. \tag{2}$$

For a model that is deterministic (i.e. has fixed parameters), this held-out loss (up to the constant factor) equals

$$\mathrm{h}[\mathbf{y}^{\text{ho}} \mid \mathbf{x}^{\text{ho}}, \mathcal{D}_{\text{t}}] - \mathrm{h}[\mathbf{y}^{\text{ho}} \mid \mathbf{x}^{\text{ho}}, \mathcal{D}_{\text{t}} \cup (x, y)]. \qquad \text{(PIC-1)} \tag{3}$$

We term this the Predictive Information Content (PIC). It quantifies *predictive* information: the part of the information gained by observing (x, y) that is applicable to predicting the holdout labels. It also relates to the Expected Predictive Information Gain (EPIG) recently proposed for active learning (Kirsch et al., 2021). In active learning, one selects points without access to their labels, and therefore the authors of Kirsch et al. (2021) must compute the expectation of (3) over $\mathrm{p}(Y \mid x, \mathcal{D}_t)$. However, here we quantify not the expected but the actual information gained when observing (x, y). We therefore follow naming conventions and term (3) a (predictive) information content. See Appendix B for more details on the relationship to active learning and EPIG.

Deriving a tractable selection function: the irreducible and reducible losses. The two terms in (3) cannot be computed efficiently; they require training on each potential point (x, y) and performing a forward pass on the full holdout set. We derive an alternative, more tractable expression. To make precise claims, we assume a Bayesian context. In the Bayesian context, we treat model parameters as a random variable with prior $\mathrm{p}(\theta)$ and infer a posterior $\mathrm{p}(\theta|\mathcal{D}_{\text{t}})$ using the sequence of already-seen training data $\mathcal{D}_{\text{t}} = b_{1:t-1}$. The model has a predictive distribution $\mathrm{p}(y|x, \mathcal{D}_{\text{t}}) = \int_\theta \mathrm{p}(y|x, \theta)\, \mathrm{p}(\theta|\mathcal{D}_{\text{t}}) d\theta$. Applying Bayes rule to $\mathrm{p}(\mathbf{y}^{\text{ho}} \mid y, \mathbf{x}^{\text{ho}}, x, \mathcal{D}_{\text{t}})$ and using the conditional independence assumption $\mathrm{p}(y_i \mid x_i, x_j, \mathcal{D}_{\text{t}}) = \mathrm{p}(y_i \mid x_i, \mathcal{D}_{\text{t}})$, we can reformulate PIC as

$$\mathrm{h}[y \mid x, \mathcal{D}_{\text{t}}] - \mathrm{h}[y \mid x, \mathbf{y}^{\text{ho}}, \mathbf{x}^{\text{ho}}, \mathcal{D}_{\text{t}}], \qquad \text{(PIC-2)} \tag{4}$$

moving the holdout set to the conditional. We show this derivation in Appendix A.

Note that the first term is simply the cross-entropy loss on the potential point (x, y) to select of a model trained on $\mathcal{D}_{\text{t}}$. The second term is the cross-entropy loss of a model trained on $\mathcal{D}_{\text{t}}$ *and the held-out set*, which can be expensive in practice as it requires constant updating with increasing $\mathcal{D}_{\text{t}}$. We can approximate this second term by training a model only on the held-out dataset, and refer to it as the *irreducible held-out loss*:

$$\mathrm{h}[y \mid x, \mathbf{y}^{\text{ho}}, \mathbf{x}^{\text{ho}}]. \qquad \text{(irreducible held-out loss)} \tag{5}$$

This approximation is reasonable when $|\mathcal{D}_{\text{t}}| \ll |\mathcal{D}_{\text{ho}}|$ or when a small model $\text{p}(y \mid \mathbf{y}^{\text{ho}}, \mathbf{x}^{\text{ho}})$ trained on $\mathcal{D}_{\text{ho}}$ has limited capacity to improve further by training on $\mathcal{D}_{\text{t}}$. Empirically, we find that this approximation does not degrade performance. Equation (5) is referred to as the irreducible held-out loss since it is the loss that could not be reduced by training on the held-out set. Plugging this approximation into equation (4), we get a new expression that we call the *reducible held-out loss*,

$$\text{h}[y \mid x, \mathcal{D}_{\text{t}}] - \text{h}[y \mid x, \mathbf{y}^{\text{ho}}, \mathbf{x}^{\text{ho}}]. \qquad \text{(reducible held-out loss)} \quad (6)$$

This provides an easy-to-compute selection function that approximates the Predictive Information Content about $\mathcal{D}_{\text{ho}}$ of the datapoint (x, y) for a model trained on $\mathcal{D}_{\text{t}}$.

Selecting informative datapoints. In this section, we have shown that training on points with higher reducible held-out loss allows a greater reduction in loss on the held-out data. In the following, we provide an intuitive perspective on why the reducible held-out loss tracks informativity. The first term, the loss $\text{h}[y \mid x, \mathcal{D}_{\text{t}}]$, is high on points that are difficult for the current model, points that are not yet learned. This is why conventional online batch selection methods use this as an selection function (Loshchilov & Hutter, 2015). However, such points might not be learnable, for example because they are noisy or mislabelled. In Figure 1 (right), we show that the plain loss disproportionally selects such noisy points. Since they are difficult also for a fully trained model, the irreducible held-out loss is also high on such points. Indeed, Figure 1 (right) shows that using low irreducible loss as the selection function allows avoiding noisy points. The reducible loss is highest on points that are difficult for the current model but not for a fully trained model—points that are learnable but not yet learned. This makes it select fewer noisy points than through uniform selection (Figure 1 right). Notably, the irreducible held-out loss identifies noisy points effectively even with a small CNN model which has 21x fewer parameters than the model used on the training set, making it cheap to train.

We point out a further potential failure mode of loss-based selection that has, to our knowledge, not been raised before. The loss is likely to be higher for outliers—values of x far from the training data, such as those in a region with low input density $\text{p}_{\text{true}}(x)$. But since our aim is to reduce the *expected* loss $\mathbb{E}_{\text{p}_{\text{true}}(X,Y)}\left[\text{h}[y \mid x, \mathcal{D}_{\text{t}}]\right]$ under $\text{p}_{\text{true}}(X, Y)$, we do not want to prioritize less relevant outliers. Since such points are also less likely in the held-out data, the irreducible loss can also be expected to be higher. Thus, the reducible held-out loss, should correct this problem and select points that are more 'relevant' to the holdout set. We find empirical evidence for this below.

Batch selection. Note that we developed the theory assuming the selection of a single point (x, y). When selecting entire batches, we select the points with the top-n_b scores from a randomly sampled set B_t of candidate examples. This assumes that each point has little effect on the scores of other points, and is often used as a simplifying assumption for selection in the context of active learning (Kirsch et al., 2019). This is a much more reasonable assumption in our case than in active learning, because model predictions are not changed much by a single gradient step on one mini-batch. Algorithm 1 shows the implementation of batch selection using the reducible held-out loss. Note that we precompute and store the irreducible losses before training. This way, we can reuse them in every epoch and even in different training runs that are executed as part of a given deep learning project.

Simple parallelized selection. For large-scale, commercially valuable neural network training, practitioners with sufficient resources would use many more machines if it further sped up training. However, as more workers are added in synchronous or asynchronous gradient descent, the returns diminish to a point where adding more workers does not further improve wall clock time (Anil et al., 2018). For example, there are rapidly diminishing returns for using larger batch sizes or distributing a given batch across more workers (LeCun et al., 2012; Keskar et al., 2016). The same holds for distributing the model across more workers along its width or depth (Rasley et al., 2020; Shoeybi et al., 2019; Huang et al., 2019). However, we can circumvent these diminishing returns by adding a new dimension of parallelization, namely, for data selection.

Since parallel *forward* passes do not suffer from such diminishing returns, one can easily use extra workers for them. The theoretical runtime speedup can be calculated as follows. The computational cost of selection is $\frac{n_b}{3n_B}$ times as much as the cost of a training step on b_t. This is because forward passes require 3x less computation than a full forward-backward pass. However, one can reduce the runtime of this computation almost arbitrarily by adding more workers that compute losses using a copy of the model being trained. The limit is reached when the time for selection is dominated by the

Algorithm 1 Reducible held-out loss selection (RHOLS)

Input: Learning rate η, model $\mathrm{p}(y \mid x, \mathcal{D}_{\text{ho}})$ trained on a holdout set $\mathcal{D}_{\text{ho}}$, batch size n_b, large batch size $n_B > n_b$.

for (x_i, y_i) in `training set` **do**
  `IrreducibleLoss[i]` $\leftarrow \mathrm{h}[y_i \mid x_i, \mathcal{D}_{\text{ho}}])$
end for

Initialize θ^0
for $t = 1, 2, \ldots$ **do**
  Randomly select a large batch B_t of size n_B.
  $\forall i \in B_t$, compute `Loss[i]`, the loss of point i given parameters θ^t
  $\forall i \in B_t$, compute `ReducibleLoss[i]`$\leftarrow$ `Loss[i]` $-$ `IrreducibleLoss[i]`
  $b_t \leftarrow$ top-n_b samples in B_t in terms of `ReducibleLoss`.
  $g_t \leftarrow$ mini-batch gradient on b_t for θ^t
  $\theta^{t+1} \leftarrow \theta^t - \eta g_t$

cost of communicating parameter updates to workers. More sophisticated parallelization strategies allow reducing the time overhead even further, as discussed in Section 4. To avoid assumptions about the particular strategy used, we report experiment results in terms of the required number of steps.

4 Related work

Data selection functions. Our reducible loss is best understood as an alternative to other selection functions. Such metrics can be categorized by the properties of points they select and by whether they use information about labels. Both high loss (Loshchilov & Hutter, 2015; Kawaguchi & Lu, 2020; Jiang et al., 2019) and low model confidence (Settles, 2009; Li & Sethi, 2006; Gal et al., 2017; Coleman et al., 2020) select for points that are "difficult" for the current model. But while loss is only applicable to labelled data, confidence is designed for the active learning setting where labels are not available. In spite of this difference, their similarity can make them face the same problems: high loss and low confidence can be due to an ambiguous ground-truth or a noisy label. Overrepresenting such noisy datapoints, as we show later, can be detrimental for training. A related technique prioritizes samples whose labels are easily forgotten during training (Toneva et al., 2018).

Variance reduction methods. Online sample selection methods have also been suggested as a way to reduce the variance of the gradient estimator computed on mini-batches (Katharopoulos & Fleuret, 2018; 2017; Johnson & Guestrin, 2018; Alain et al., 2015). Such methods typically use importance sampling, where points with high gradient norm (or approximations thereof) are sampled with high probability. To avoid biasing the gradient estimator, the gradient of high-probability points is then down-weighted. Importance sampling can be effective but also sensitive to hyperparameters (Johnson & Guestrin, 2018). Furthermore, Figure 1 (right) shows that the gradient norm is also often highest for noisy points, which may make it unsuitable for noisy real-world datasets.

Computationally efficient data selection. While we limit our analysis to the choice of selection function and compute the metrics naively, this choice is likely inefficient in practice. Selection can be made cheaper by reusing losses computed in previous epochs (Loshchilov & Hutter, 2015; Jiang et al., 2019) or training a small model to predict them (Katharopoulos & Fleuret, 2017; Zhang et al., 2019). Alternatively, core set methods only perform selection once before training (Mirzasoleiman et al., 2020; Borsos et al., 2020). Such methods typically use expensive metrics that are not economical for large web scraped datasets where training typically only lasts a few epochs. However, we compare to a recent method that reduces selection costs by using a smaller proxy model (Coleman et al., 2020).

Time-efficient data selection. To accelerate data selection, prior work uses a set of workers that perform forward passes on B_t or on the entire dataset asynchronously while the master process trains on recently selected data (Alain et al., 2015; Schaul et al., 2015). This supplements rather than replaces standard data parallelism, a method whose returns diminish as more workers are added

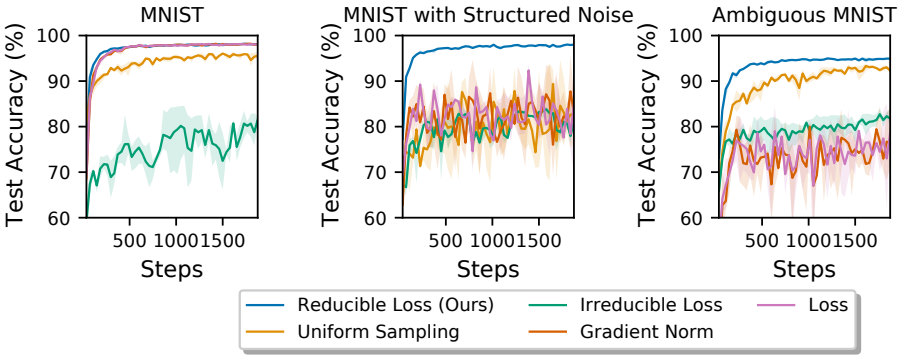

Figure 2: Reducible loss is robust to a variety of label noise patterns, while every other selection method degrades. Lines correspond to means and shaded areas to minima and maxima across 3 random seeds.

(Anil et al., 2018). Furthermore, forward passes can be accelerated using low-precision hardware. A forward pass by default already requires roughly 3x less time than a forward-backward pass but this speedup can be increased to a factor around 10 when using the low precision cores available in modern GPUs and TPUs (Jouppi et al., 2017; Jiang et al., 2019). While backward passes typically require higher precision, forward passes can tolerate lower precision.

5 Experiments

In this section, we evaluate how much selecting batches through our selection function accelerates deep neural networks training. We compare to several recent alternatives.

Datasets. We evaluate on four benchmarks of increasing scale. 1) QMNIST (Yadav & Bottou, 2019) uses the MNIST training set (LeCun et al., 1998) and 50,000 additional images that serve as a slightly more difficult test set. By using this as the holdout set, we can use the original MNIST test set with 10,000 images as the holdout set. 2) On CIFAR-10 (Krizhevsky & Hinton, 2009) we train on half of the training set and use the other half as a holdout set used for training the irreducible loss model. 3) CIFAR-100: same as CIFAR-10. 4) CINIC-10 (Darlow et al., 2018) has 4.5x more images than CIFAR-100 and includes a holdout set and a test set with 90,000 images each.

In settings where training time, not data, is the critical constraint, one can often reserve some data $\mathcal{D}_{\text{ho}}$ to compute (6). Indeed, for massive web scraped datasets, where training time is most critical, there is commonly not enough time to train for even one full epoch (Brown et al., 2020; Komatsuzaki, 2019; Kaplan et al., 2020). Note that this is not the case for our benchmarks, meaning we have to reserve holdout data in the manner described above.

Baselines. The most important baseline is the *de facto* standard in deep learning, uniform data sampling (without replacement). We also evaluate against selection functions that have achieved competitive performance recently: the plain cross-entropy loss (as implemented by Kawaguchi & Lu (2020)), gradient norm, and gradient norm with importance sampling (called *gradient norm IS* in our figures) (Katharopoulos & Fleuret, 2018). As the per-example gradient norm is prohibitive to compute, we follow prior works in approximating it with the norm of the gradient for the last layer. We also compare to a recent method (SVP (Coleman et al., 2020)) that selects data offline before training in order to probe if online selection is needed for maximal performance. We report results of the maximum entropy selection function of this method, and select with the best-performing model, ResNet18. We compare only to alternative selection functions. Other improvements such as predicting or reusing losses (Katharopoulos & Fleuret, 2017; Zhang et al., 2019; Jiang et al., 2019) are orthogonal to our contribution; in practice, they can and should be combined with our selection function to make selection cheaper.

Experiment details. For experiments on QMNIST, we use a multi-layer perceptron with 2 hidden layers and 512 units in each hidden layer. For experiments on CIFAR-10, CIFAR-100, and CINIC-10,

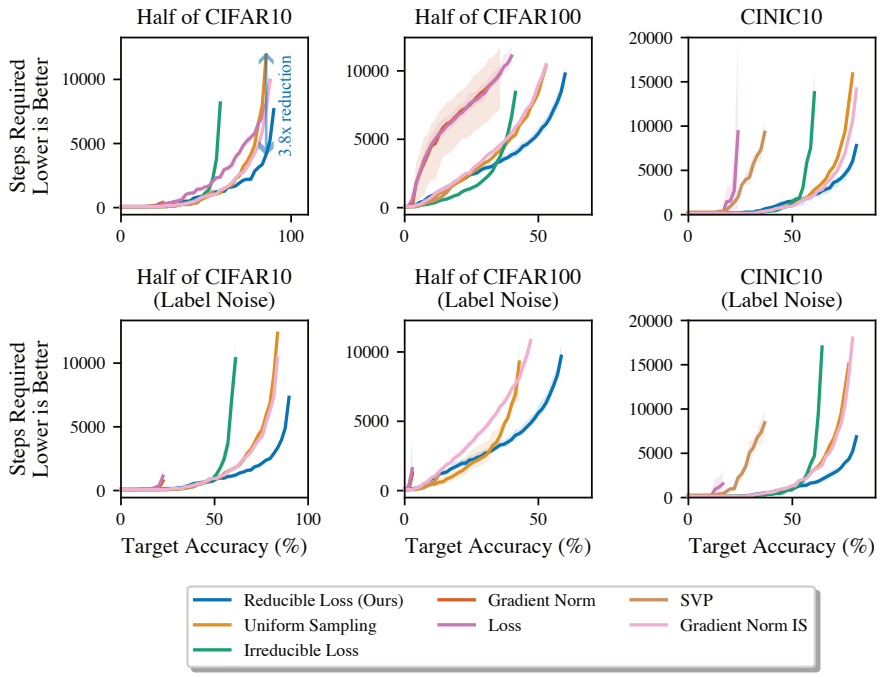

Figure 3: **Top:** Gradient steps required to achieve a given test accuracy (lower is better). **Bottom:** As before, but with 10% uniform label noise applied. Lines correspond to means and shaded areas to minima and maxima across 3 random seeds.

we use a ResNet-18. All models are trained using the Adam optimizer with default hyperparameters (β_1=0.9, β_2=0.999, and weight decay of 0.01), learning rate 0.001, a training batch size $n_b = 32$ (64 for CINIC-10) with $n_B = 320$ (640 for CINIC-10) points pre-sampled, meaning we select $\frac{n_b}{n_B} = 10\%$ of points. On all datasets except QMNIST, we train using data augmentation (random crop and horizontal flip). The irreducible loss models are trained on holdout sets (not test sets, see above). For each dataset, we select the irreducible loss model from the epoch with lowest held-out loss, determined via cross-validation. This model is then used for online batch selection.

5.1 Robustness to noise

We begin by evaluating the performance of different selection methods under a variety of noise patterns on QMNIST and variations thereof. We use this dataset because of the low rate of label noise in its original form, allowing us to test the effect of adding noise. Firstly, we add uniform label noise to 10% of training points. Secondly, we add structured label noise that affects easily confused classes. We follow Rolnick et al. (2017) and flip the labels of the four most frequently confused classes (in the confusion matrix of a trained model) with 50% probability. For example, a 2 is often confused with a 5; thus we change the label of all 2s to 5s with 50% probability. Third, we leverage the natural noise distribution of MNIST by using AmbiguousMNIST (Mukhoti et al., 2021) as the training set. AmbiguousMNIST contains a training set with 60k generated ambiguous digits that have more than one plausible label.

While selecting with loss and gradient norm trains stably on the MNIST training set, their performance consistently degrades on each type of noise distribution (Figure 2). This is in line with our earlier finding that they over-select points with noisy labels. In contrast, reducible held-out loss is robust to label noise and trains in fewer steps than all baselines.

5.2 Speedup in steps required

Here, we evaluate how many steps each selection method needs to achieve a given test accuracy.

As our focus is on evaluating a new selection function rather than an entire training pipeline, we report the number of steps needed to reach a target accuracy. How the required steps relate to the wall clock time depends on implementation details that are beyond the scope of this paper. Most importantly, data selection is amenable to parallelization beyond standard data parallelism as discussed in Sections 3 and 4. Furthermore, a model trained on a holdout set can be reused for selection during the many runs that are part of a training pipeline.

Speedup on clean data. RHOLS reaches the same accuracies in fewer steps than standard uniform selection without replacement across all datasets (Figure 3, top). Furthermore, it outperforms all baselines across all datasets. The observed speedup with respect to uniform selection is 3.82x on CIFAR-10, 1.76x on CIFAR-100, and 2.88x on CINIC-10.

Speedup on noisy data. Since the most expensive training runs in machine learning are typically on noisy web scraped data, it is particularly interesting to see how the investigated methods are affected when moderate amounts of label noise are added. In the following experiments, we uniformly perturb the labels for 10% of the datapoints.

Under label noise, batch selection with reducible held-out loss further increases its speedup over the other methods (Figure 3, bottom). Particularly interesting is that the speedup over uniform selection also grows on noisier data. The speedups we observe relative to uniform selection are 5.15x on CIFAR-10, 2.34x on CIFAR-100, and 4.17x on CINIC-10. This aligns with our information-theoretical perspective on reducible held-out loss and with the finding in Figure 1 (right) that reducible held-out loss selects fewer noisy points compared to uniform selection, which itself selects fewer than other methods.

Reducible held-out loss can select more relevant points. In Section 3, we argued that, in contrast to other methods, RHOLS does not prioritize less relevant points that have low probability under the input density $\mathrm{p}_{\text{true}}(x)$. To study this empirically, we need to manipulate $\mathrm{p}_{\text{true}}(x)$ since we cannot measure the density $\mathrm{p}_{\text{true}}(x)$ of selected points directly. This is intended to illustrate that the reducible loss selects points more relevant to the holdout set, with or without manipulation. To investigate this, we subset the CINIC10 dataset; the train set consists of all classes, while the holdout and test set consists of half of the classes. Such a setup could also be useful in some real-world scenarios where one would like to avoid certain training points containing e.g. low-quality, biased, or toxic information. This could be done by training the holdout model on a clean holdout set.

In this scenario, we find that RHOLS avoids points whose labels do not appear in the validation set (Figure 4). Accordingly, it provides a large speedup over baselines. Irreducible loss selection is also very effective at this, as expected, while other baselines select these points half of the time. Compared to irreducible loss, RHOLS selects points that are additionally not yet learned, leading to a greater speedup. This suggests that RHOLS is able to identify training examples that are relevant for the task at hand. The holdout data implicitly provides a task relevance description that RHOLS exploits.

5.3 Ablations

Irreducible loss model. In the sections above, we have used irreducible loss models with the same architecture and size as the main model. The cost for training this model can be amortized over the many training runs typically performed in a machine learning project—runs for tuning hyperparameters, evaluating on multiple seeds, ensembling, or investigating the impact of various design choices.

In addition, we demonstrate on CIFAR-10 that training the irreducible loss model does not have to be expensive. We test 5 different irreducible loss models which were trained for 1, 2, 4, 7, and 91 epochs, corresponding to an equal partition of the space from worst (1 epoch) to best (91 epochs) validation set accuracy. The model trained for 7 epochs gives a similar speed up in training steps as the model with the lowest validation loss (Fig. 5). In fact, the irreducible loss model need not have the same architecture as the main model, and can be considerably smaller. On CIFAR-10, we use a small 1990s-style CNN irreducible loss model that has 21x fewer parameters than the ResNet-18 (0.54M vs 11.17M) and uses 29x fewer FLOPs per forward pass. Using various versions of this small model for batch selection results in comparable or even greater speedups vs uniform than using a Resnet-18 (Figure 5).

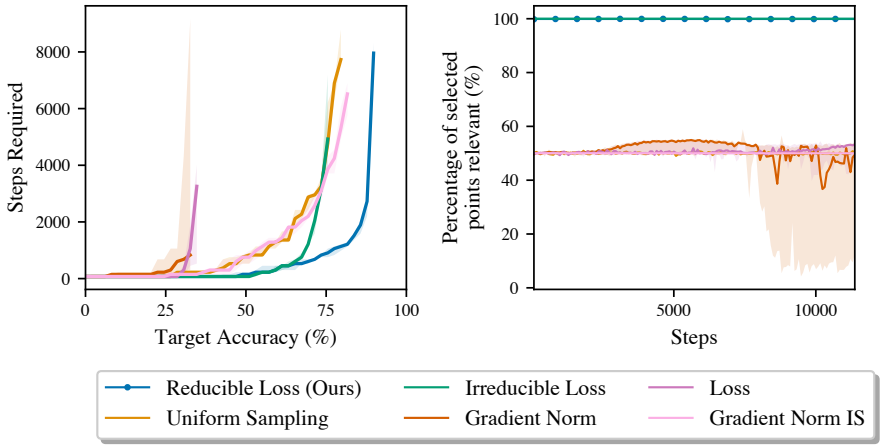

Figure 4: CINIC-10 with half of the classes absent (irrelevant) in the test and holdout sets. **Left:** steps required to reach a given target accuracy on the test set. **Right:** Proportion of selected points whose labels do appear in the holdout and test sets. RHOLS and irreducible loss select exclusively the relevant classes. ROHLS provides a large speedup over existing baselines. Lines correspond to means and shaded areas to standard deviations across 3 random seeds

Varying selectivity. Our method has a hyperparameter, the percentage $\frac{n_b}{n_B}$ of evaluated points which are selected for training. In the experiments above, this parameter was set to 0.1. We have not tuned this parameter, as we aim to analyse how well our method works "out of the box". In fact, on some datasets, performance further improves with other values of this parameter (Appendix C).

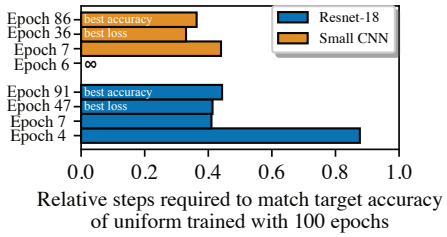

Figure 5: Speedup in training steps vs. uniform selection, when using different irreducible loss models on CIFAR-10. An irreducible loss model trained for only 7 epochs, or a 21x smaller CNN, achieve similar speedups. Average over 3 random seeds.

6 Discussion

With ever larger models and datasets, training times keep low-resourced groups from access to deep learning and well-resourced groups from pushing its frontier. This can already be mitigated by existing methods for parallelization and data selection. However, the benefits are limited by diminishing returns for the former and, as we show, problems with noisy data for the latter. We aim to alleviate these limits by introducing a novel, theoretically grounded selection function. In controlled experiments, we show that its deployment enables substantial speedups on curated data and even larger ones under label noise.

Despite good performance on the investigated datasets, we make no claim that our particular implementation of the novel selection function reduces wall clock times in all practical settings. Looking forward, it is important to scrutinize and reduce the computational cost of the selection process. We see promising avenues in predicting the losses (Zhang et al., 2019; Katharopoulos & Fleuret, 2017), reusing previously computed ones (Loshchilov & Hutter, 2015; Schaul et al., 2015), or using a smaller proxy model for estimating them (Coleman et al., 2020; Zhang et al., 2019), e.g. by distilling the main model into the smaller model online (Anil et al., 2018). Additionally, data selection should be run in parallel and executed on fast low-precision cores for maximal speedups.

References

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Zalán Borsos, Mojmír Mutnỳ, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *arXiv preprint arXiv:2006.03875*, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *International Conference on Learning Representations*, 2020.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On statistical bias in active learning: How and when to fix it. *arXiv preprint arXiv:2101.11665*, 2021.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192. PMLR, 2017.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32: 103–112, 2019.

Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31:7265–7275, 2018.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Angelos Katharopoulos and François Fleuret. Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*, 2017.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018.

Kenji Kawaguchi and Haihao Lu. Ordered sgd: A new stochastic optimization framework for empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 669–679. PMLR, 2020.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *arXiv preprint arXiv:1906.08158*, 2019.

Andreas Kirsch, Tom Rainforth, and Yarin Gal. Active learning under pool set distribution shift and noisy data. *CoRR*, abs/2106.11719, 2021. URL `https://arxiv.org/abs/2106.11719`.

Aran Komatsuzaki. One epoch is all you need. *arXiv preprint arXiv:1906.06669*, 2019.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

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

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Mingkun Li and Ishwar K Sethi. Confidence-based active learning. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1251–1261, 2006.

Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020.

Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506, 2020.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Burr Settles. Active learning literature survey. 2009.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Yonglong Tian, Olivier J Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*, 2021.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Chhavi Yadav and Léon Bottou. Cold case: The lost mnist digits. In *Advances in Neural Information Processing Systems 32*, 2019.

Jiong Zhang, Hsiang-Fu Yu, and Inderjit S Dhillon. Autoassist: A framework to accelerate training of deep neural networks. *arXiv preprint arXiv:1905.03381*, 2019.