Prioritized training on points that are learnable, worth learning, and not yet learned

Sören Mindermann [* 1] **Muhammed Razzak** [* 1] **Winnie Xu** [* 2 3] **Andreas Kirsch** [1] **Mrinank Sharma** [4]
Adrien Morisot [3] **Aidan N. Gomez** [1 3] **Sebastian Farquhar** [1] **Jan Brauner** [1] **Yarin Gal** [1]

Abstract

We introduce *Goldilocks Selection, a technique for faster model training which selects a sequence of training points that are "just right"*. We propose an information-theoretic acquisition function—the reducible validation loss—and compute it with a small proxy model—GoldiProx—to efficiently choose training points that maximize information about the labels of a validation set. We show that the "hard" (e.g. high loss) points usually selected in the optimization literature are typically noisy, while the "easy" (e.g. low noise) samples often prioritized for curriculum learning confer less information. Further, points with uncertain labels, typically targeted by active learning, tend to be less relevant to the task. In contrast, *GoldiProx Selection* chooses points that are "just right" and empirically outperforms the above approaches. Moreover, the selected sequence can transfer to other architectures; practitioners can share and reuse it without the need to recreate it.

1. Introduction

Models such as GPT-3 (Brown et al., 2020), CLIP (Radford et al., 2021), and ViT (Dosovitskiy et al., 2021) leverage vast quantities of parameters and data to produce remarkable results. Yet, with great size come great computation costs and prohibitive training times. Accounting further for hyperparameter selection and architecture choice, compute is the core bottleneck that limits performance. Improving computational efficiency paves the way for larger, better performing models.

Practitioners have long realized that not all samples in the

---
[*]Equal contribution [1]OATML, University of Oxford, United Kingdom [2]Department of Computer Science, University of Toronto, Canada [3]Cohere [4]Department of Statistics, University of Oxford, United Kingdom. Correspondence to: Sören Mindermann <soeren.mindermann@gmail.com>.

largest datasets are equally useful. Many samples are *noisy*, i.e. mislabelled or inherently ambiguous, so that their label is unpredictable. For example, the next word in a sentence often cannot be foreseen and the caption of 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). These samples are quickly learned and most could then be ignored without losing performance. Fortunately, data is often so abundant that we can hardly finish a single epoch with the available compute (Brown et al., 2020; Kaplan et al., 2020), meaning that we can easily afford to skip some points.

In the optimization literature, online batch selection methods (Loshchilov & Hutter, 2015) train only on points that are 'hard' to the current model (e.g. high loss or high gradient norm). These techniques include both unbiased importance sampling methods (Katharopoulos & Fleuret, 2018), including prioritized experience replay in reinforcement learning (Schaul et al., 2015), as well as biased methods (Kawaguchi & Lu, 2020) which simply train on the top-k points with the highest loss or gradient norm. Although online selection can be costly since it computes a forward pass on every point considered, it successfully skips redundant points.

First, we demonstrate that prioritising hard, high-loss examples fails on noisy data. In real-world data, high-loss examples may be mislabelled or inherently ambiguous. We find that, with as little as 10% uniform label noise, high loss points are overwhelmingly those where the label noise is realized. Prioritising them degrades performance severely.

Instead of prioritizing hard points, other bodies of literature including curriculum learning prioritize "easy" points with low noise before training on all points equally (Bengio et al., 2009). While this can improve training and convergence, it fails to prioritize points that are not redundant.

A third body of work, active learning, often selects points whose labels are uncertain to the model (Houlsby et al., 2011; Gal et al., 2017). This approach can choose informative points while also avoiding noisy ones. However, we show that it has another weakness: selecting the *less rele-*

vant points that are unlikely to appear at test time, such as those with a low input density $p(x)$. These are unfamiliar to the model and can therefore be selected by active learning approaches. Favouring such obscure points is problematic since they are abundant in uncurated datasets.

To overcome these limitations, we introduce *Goldilocks Selection*. Building on recent progress in active learning (anonymous), we introduce an information-theoretic acquisition function—pointwise predictive information gain—which selects points that are "just right". By maximizing the information gained about the labels of the validation set, our method chooses non-redundant points without preferring noisy or irrelevant points. Furthermore, we show its equivalence to an intuitive and easy-to-implement novel acquisition function: reducible held-out cross-entropy loss.

Additionally, by using a small proxy model, *GoldiProx Selection* performs online batch selection efficiently. We perform the forward pass for batch selection using a smaller, less compute-intensive model, thereby reducing the computational overhead. Nevertheless, since the proxy sees the same data in the same order as the primary learner, its information state matches the primary model. As such, we find the two models select similar data.

Furthermore, we find that the sequence of training data selected by one small proxy—the GoldiProx Sequence—can accelerate training for models of different size *and* architecture. As such, practitioners can select a sequence one time and reuse it for many training runs, or even download a data sequence created by someone else.

2. Background: Online batch selection

Consider training a model $p(y \mid x, \theta)$ 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 $|b|$ 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, we pre-sample a larger batch B_t of size $|B| > |b|$ and rank the points x_i in this batch according to a label-aware acquisition function $A(x_i, y_i)$. Then, we construct a smaller batch b_t that consists of the top-ranking $|b|$ points in B_t and perform one gradient step to minimize a mini-batch loss $\sum_{i \in b_t} L(y_i, p(y_i \mid x_i, \theta))$. The following large batch B_{t+1} is then pre-sampled from $\mathcal{D}$ without replacement of the previously sampled points (all points are replaced at the start of the next epoch).

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

los & Fleuret, 2018) or to include earlier points x_i in B_t whose acquisition score $A(x_i, y_i)$ 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).

3. What makes points informative?

3.1. High loss comes with high noise

Online batch selection methods typically recommend to train on points with high loss or high gradient norm. Here, we focus on loss because it is significantly easier to compute and correlates with the gradient norm (Katharopoulos & Fleuret, 2018). Intuitively, high-loss points can be thought of as 'hard' to the current model and therefore not currently redundant (intuitive concepts in this section will be formalized later).

While loss-based selection can perform well on curated datasets (Kawaguchi & Lu, 2020), we show that it underperforms when adding just 10% uniform label noise[1]. In Figure 1, we find that high-loss points (blue line) are overwhelmingly the points mislabelled by the noise distribution across three datasets: QMNIST (same training set as MNIST with a larger test set) (LeCun et al., 1998; Yadav & Bottou, 2019), CIFAR-10 (Krizhevsky & Hinton, 2009), and CINIC-10 (a subset of ImageNet 4.5x larger than CIFAR-10 (Darlow et al., 2018)). Here, training on the $|b|$ highest-loss points (Figure 2, blue) in each large batch B_t degrades performance severely compared to e.g. uniform randomly sampled data (green) and reducible loss (introduced in Section 5).

3.2. Only removing noisy points is not enough

Avoiding noisy points is recommended, amongst other things, in the literature on curriculum learning (Bengio et al., 2009). These techniques initially prioritize non-noisy points as noisy points are considered too hard. Eventually, they train on all points in $\mathcal{D}$ equally, although other methods avoid noisy points altogether (Thulasidasan et al., 2019).

To probe the effect of prioritizing non-noisy points, we train a separate model $p(y \mid x, \theta_{\text{val}})$ to convergence on a validation set $\mathcal{D}_{\text{val}}$ and then evaluate its loss on every point in $\mathcal{D}$. Since a noisy point x likely has high loss even at the end of training (if x was held out during training), we can effectively remove it by using the acquisition function $A(x, y) = -L(y, p(y \mid x, \theta_{\text{val}}))$ (Figure 1, red). However,

---

[1]We can think of a *test time* label as noisy whenever it cannot be predicted with the chosen model class even with unlimited training data (Der Kiureghian & Ditlevsen, 2009). With this definition, noise is common even when labelling is deterministic.
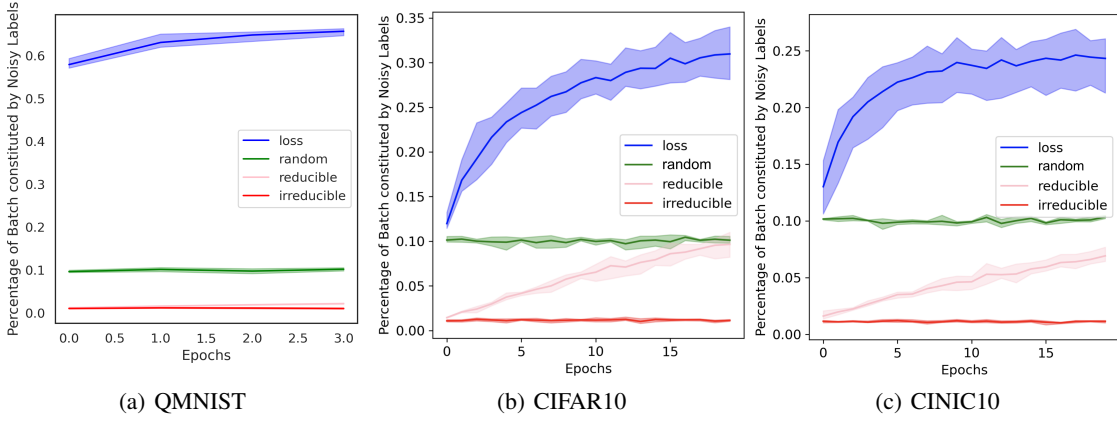
Figure 1. Percentage of selected points that are corrupted with label noise using different online batch selection criteria. Training data are corrupted with 10% uniform label noise. Mean along with minimum and maximum across 3 runs shown. Using LeNet model; low epoch setting; no data augmentation.
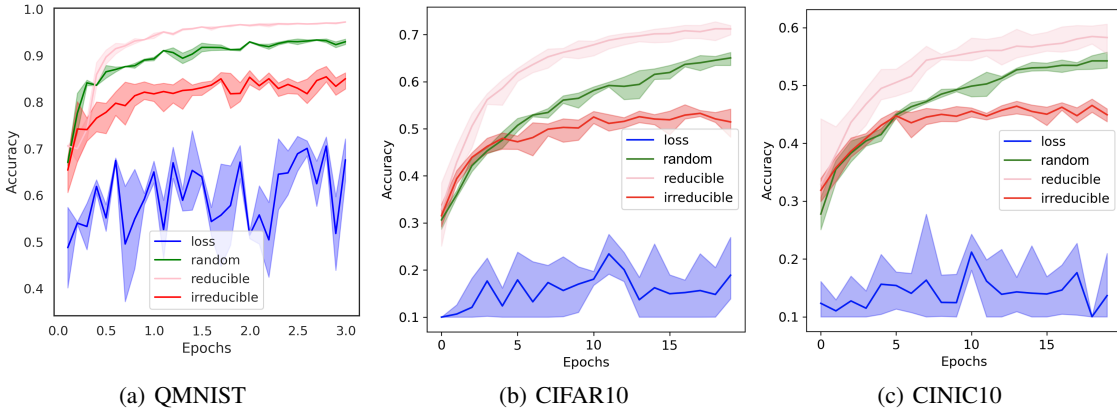

Figure 2. Image classification test set accuracy for training data with 10% label noise using using different online batch selection criteria. Mean along with minimum and maximum across 3 runs shown. Using LeNet model; low epoch setting; no data augmentation.

this does not improve test accuracy (irreducible loss in Figure 2, red) compared to random sampling (green) as it has no mechanism to avoid redundant (too easy) points.

3.3. Prioritizing information that is relevant to the prediction task

Active learning typically prioritizes points with uncertain labels[2]—either measured by a high-entropy model output or by disagreement between models (Gal et al., 2017). No access to labels is assumed. A prominent approach, Bayesian Active Learning with Disagreement (BALD), even selects informative points that are also predicted to be less noisy (Houlsby et al., 2011; Gal et al., 2017). In a Bayesian model with parameters given by the random variable Θ, it quantifies the expected information (or mutual information)

gained about Θ when observing the unknown label Y for x: $I[\Theta; Y|\mathcal{D}_t, x]$.

While observing uncertain points reduces uncertainty about the parameters, we show that it can fail to reduce uncertainty that is relevant to the prediction task. We add 20% low-relevance points to the data which have uniform white noise input x, meaning they have low input density $p(x)$. Since a low-density point is unlikely to appear at test time, it is not necessarily worth learning.

Indeed, BALD selects these points more than our method. After a warmup period needed to obtain reasonable uncertainty estimates (Gal & Ghahramani, 2016), the white noise samples constitute approximately 20% of the batches selected by BALD (matching the baseline of uniform random sampling).

BALD's behavior can be explained when we consider that more information can be gained about the parameters in

---

[2]Instead of label uncertainty, some methods select for data diversity (Brinker, 2003). Both criteria help avoid redundancy.

areas of the input space where data is still scarce. However, points in such areas have low input density and are therefore unlikely to appear at test time. BALD's tendency to select low-density points is likely to be more extreme in high-kurtosis (heavy tailed) input distributions where a significant share of points are located in low-density areas.
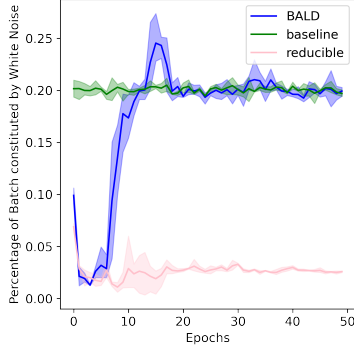

Figure 3. Percentage of selected points on QMNIST that have white noise input (low data density / less likely to appear at test time), using various online batch selection methods.

4. Experimental details

For experiments on QMNIST, we use a 3-layer MLP with 512 hidden units. For experiments on CIFAR-10 and CINIC-10, we use a small CNN, reminiscent of LeNet (LeCun et al., 1998). All models are train using the AdamW optimizer with learning rate 0.001. In Figure 3 for BALD (blue) we use 0.005 learning rate and 0.5 dropout rate. Hyperparameters are not tuned for the performance of our method. Since the goal is to reduce training steps, we evaluate without data augmentation and in the low-epoch setting. Indeed, efficiently training a large model often means training for a single epoch or less without augmentation and stopping far short of convergence (Brown et al., 2020; Kaplan et al., 2020).

Baselines: We use the Kawaguchi & Lu (2020) (high loss points) as the external baseline, in addition to random selection and irreducible loss. Some other online batch selection methods (Loshchilov & Hutter, 2015; Katharopoulos & Fleuret, 2018) are complementary to ours and could be combined with our methods in future work. We do not consider curriculum learning, coreset methods, and SVRG (Johnson & Zhang, 2013) (these typically cost too much upfront computation when training only for a few epochs, or do not aim to accelerate training) or active learning (which assumes no access to labels).

5. Pointwise predictive information gain—a measure of reducible loss

Based on the previous sections, we are motivated to prioritize points that are learnable[3] (not noisy) but difficult to the current model and relevant to the validation set. To that end, we propose an information-theoretic acquisition function that maximizes the information gained about the labels of a validation set.

We introduce our method in a Bayesian context. We treat model parameters as a random variable Θ with prior $p(\theta)$ and infer an approximate posterior $p(\theta|\mathcal{D}_t)$ using the sequence of already-seen training data $\mathcal{D}_t = b_{1:t-1}$ from the set $\mathcal{D}$. The model has a predictive distribution $p(y|x, \mathcal{D}_t) = \int_\theta p(y|x, \theta)\, p(\theta|\mathcal{D}_t)d\theta$. However, our method works well with standard non-Bayesian networks which use a point estimate of θ.

The validation set is denoted as $\mathbf{x}^{\text{val}}, \mathbf{y}^{\text{val}} := \{x_i\}_{i=1}^{n_{\text{val}}}, \{y_i\}_{i=1}^{n_{\text{val}}}$.

We would like to add a single[4] point (x, y) to $\mathcal{D}_t$ that minimizes the validation cross-entropy loss $\sum_{i \in \mathcal{D}_{\text{val}}} -\log p(y_i|x_i, \mathcal{D}_t, x, y)$. Thus, we must maximize the information gained about the validation labels. Anonymous (2021) notes that this information quantity, termed Expected Predictive Information Gain (EPIG), can select informative unlabeled points in the active learning setting. There, the label y for x is not available and so we must compute the expected information gain (also known as mutual information) between $\mathbf{Y}^{\text{val}}$ and Y:

$$\text{I}[\mathbf{Y}^{\text{val}}; Y \mid \mathbf{x}^{\text{val}}, x, \mathcal{D}_t] = H[\mathbf{Y}^{\text{val}}|\mathbf{x}^{\text{val}}, \mathcal{D}_t] \quad (1)$$
$$- H[\mathbf{Y}^{\text{val}}|\mathbf{x}^{\text{val}}, \mathcal{D}_t, x, Y], \quad (2)$$

which requires computing an expectation over the random variables Y and $\mathbf{Y}^{\text{val}}$ (respectively conditioned on x and $\mathbf{x}^{\text{val}}$).

Here, we introduce a new quantity, avoiding the need to compute expensive expectations by making use of the labels $\mathbf{y}^{\text{val}}$ and y. The label-aware version of the mutual information is known as pointwise mutual information ($\text{pmi}[\cdot]$); hence we refer to our metric as the pointwise predictive information gain:

$$\text{pmi}[\mathbf{y}^{\text{val}}; y \mid \mathbf{x}^{\text{val}}, x, \mathcal{D}_t] = h[\mathbf{y}^{\text{val}} \mid \mathbf{x}^{\text{val}}, \mathcal{D}_t] \quad (3)$$
$$- h[\mathbf{y}^{\text{val}} \mid \mathbf{x}^{\text{val}}, \mathcal{D}_t, x, y]. \quad (4)$$

The quantities above are the pointwise mutual information

---

[3]Can be learned without overfitting to them.

[4]In practice, we add a whole batch b_t. Adding multiple points at once is more complex in theory (Kirsch et al., 2019) but we can ignore this complication whenever individual training steps cause a negligible change to the model. Acquiring individual points in this case would lead to the same selections.

and pointwise conditional entropy denoted in lower case:

$$h[y \mid x] = -\log p(y \mid x); \tag{5}$$

$$\text{pmi}[y_1, y_2 \mid x_1, x_2] := -\log \frac{p(y_1 \mid x_1)\, p(y_2 \mid x_2)}{p(y_1, y_2 \mid x_1, x_2)}. \tag{6}$$

Note that $h[y \mid x]$ is just the cross-entropy loss.

This measures the desired quantity: the realized reduction in uncertainty about the validation labels $\mathbf{y}^{\text{val}}$ due to observing (x, y). As a result, it satisfies our criteria intuitively: 1) A redundant point (already learned) confers little new information about the validation labels. 2) So does observing a noisy label since a different label may have appeared if the data generation process were repeated. 3) A point that is unlikely to appear in the validation data, and is therefore not relevant to it, also confers little information about it.

Algorithm 1 GoldiProx Selection

1: **Input:** Initial parameters θ_{small}^0 and θ_{big}^0, learning rate η, small model $p(y \mid x, \theta_{\mathbf{val}})$ trained on validation set, batch size $|b|$, large batch size $|B| > |b|$.

2: **for** i in `training set` **do**
3: $\quad$ `IrreducibleLoss[i]` $\leftarrow L(y_i, p(y_i | x_i, \theta_{\mathbf{val}}))$
4: **end for**
5: `Sequence` $\leftarrow []$

6: **for** $t = 1, 2, \ldots$ **do** $\qquad \triangleright$ Select data with small model
7: $\quad$ Randomly select a large batch B_t of size $|B|$.
8: $\quad$ $\forall i \in B_t$, compute `Loss[i]`, the loss of point i given parameters θ_{small}^t
9: $\quad$ $\forall i \in B_t$, compute `ReducibleLoss[i]` $\leftarrow$ `Loss[i]` $-$ `IrreducibleLoss[i]`
10: $\quad$ $b_t \leftarrow$ top-$|b|$ samples in B_t in terms of `ReducibleLoss`.
11: $\quad$ $g_t \leftarrow$ mini-batch gradient on b_t for θ_{small}^t
12: $\quad$ $\theta_{\text{small}}^{t+1} \leftarrow \theta_{\text{small}}^t - \eta g_t$
13: $\quad$ Append b_t to `Sequence`.
14: **end for**
15:
16: **for** $t = 1, 2, \ldots$ **do** $\qquad\qquad \triangleright$ Train large model
17: $\quad$ Load $b_t \leftarrow$ `Sequence[t]`
18: $\quad$ $g_t \leftarrow$ mini-batch gradient on b_t for θ_{big}^t
19: $\quad$ $\theta_{\text{big}}^{t+1} \leftarrow \theta_{\text{big}}^t - \eta g_t$
20: **end for**

However, the two terms in (3) are still inefficient to compute, as they require training on (x, y) and performing a forward pass on the full validation set. We rewrite (3) using the symmetry of (pointwise) mutual information, giving:

$$h[y \mid x, \mathcal{D}_t] - h[y \mid x, \mathbf{x}^{\text{val}}, \mathbf{y}^{\text{val}}, \mathcal{D}_t]. \tag{7}$$

The first term is simply the cross-entropy loss for x of the current model trained on D_t. The second term is the cross-

entropy loss of a model trained on D_t and the validation set. Both can be practically computed but we can approximate and further simplify to

$$h[y \mid x, \mathcal{D}_t] - h[y \mid x, \mathbf{x}^{\text{val}}, \mathbf{y}^{\text{val}}]. \tag{8}$$

This approximation is reasonable when $\mathcal{D}_t \ll \mathcal{D}_{\text{val}}$, or when the model $p(y \mid \mathbf{y}^{\text{val}})$ trained on $\mathcal{D}_{\text{val}}$ has limited capacity to fit the additional data $\mathcal{D}_t$ (see next section).

The second term in (8) is simply the acquisition function we introduced in section 3.2—the irreducible validation loss. This makes (8) an intuitive and theoretically grounded metric: the reducible held-out cross-entropy loss.

6. Using a small proxy model to select data for other models online

Computational savings from online batch selection have a limit. Compute is only saved by avoiding backward passes. For every point we consider selecting, we need to compute a forward pass to evaluate its acquisition function $A(x_i, y_i)$ (Kawaguchi & Lu, 2020; Katharopoulos & Fleuret, 2018). Although the forward pass is easy to parallelize and therefore time-efficient, it is not compute-efficient.

The high cost of data selection has also been studied in the literature on active learning, where Coleman et al. (2020) proposed to select points that are informative as indicated by a fixed small proxy model. This can save drastically more compute, especially when the main model is very large. We build on this approach, extending it to the setting where labels are available and data is selected online rather than with a fixed model.

In the online setting, we must ensure that, at step t, the proxy model $p(y|x, \theta_{\text{proxy}})$ selects points informative to the larger model given the large model's information state (described by $\mathcal{D}_t$). Their information states should not differ much. For example, a point that is already learned by a converged model (converged on all of $\mathcal{D}$) may still be informative to a model trained on the sequence $\mathcal{D}_t$ if $x \in \mathcal{D} \setminus \mathcal{D}_t$).

Matching occurs naturally if we store the selected *sequence* of points, not just the set, as shown in Algorithm 1. First, we fully train the proxy using standard online batch selection as described in Section 2. We append each selected batch b_t to the data sequence $\mathcal{D}_t$ (storing only the indices i of x_i for space efficiency). Subsequently (or simultaneously), we train the large model on $\mathcal{D}_t$ in the stored order.

Matched training on a sequence is simple to implement. For example, one practitioner can create the sequence $\mathcal{D}_t$ and another can use it as a plug-in replacement for their existing data loader without changing their training setup.

Empirically, the two matched models do select similar points as indicated by positive rank correlations (Figure 4, blue).
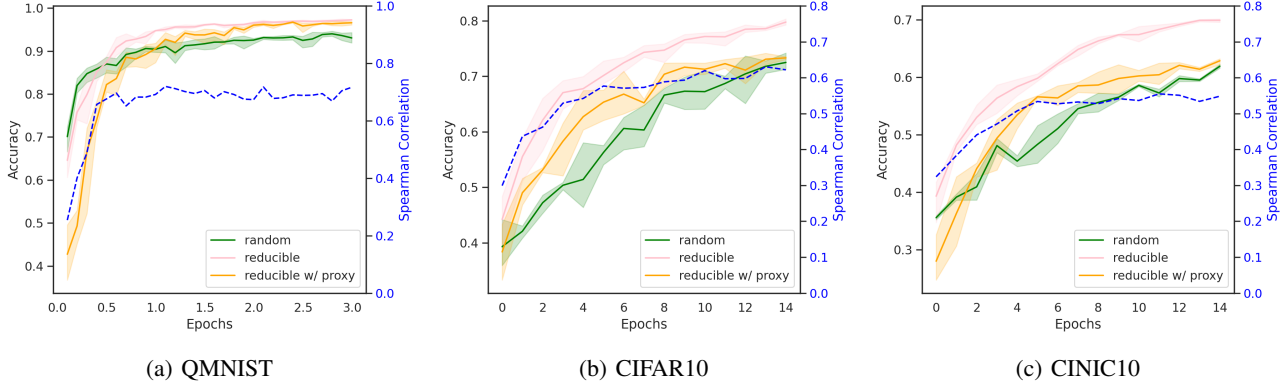
(a) QMNIST   (b) CIFAR10   (c) CINIC10

Figure 4. **Transfer to larger model with different architecture // rank correlations.** Test set accuracy for image classification tasks using reducible loss with a small proxy and transferring the selected GoldiProx sequence to a large model with different architecture that uses 29x more FLOPs (7x on QMNIST). Methods: Reducible loss (GoldiProx Selection) with and w/o proxy, and random selection. Right axis: rank correlation between reducible loss computed with small proxy model and large model. Positive correlations indicate similar selections.

Note that perfect correlations are possible even if the larger model outperforms the proxy across all points.

6.1. Experiments

We illustrate the effectiveness of our method on QMNIST, CIFAR-10 and CINIC-10.

On QMNIST, we demonstrate that we can use a similar model with small capacity as a proxy model for a large model. We train a 3-layer MLP with 128 hidden units as our small proxy model, and use a 3-layer MLP with 512 hidden units as our large model. This larger MLP has **6.93×** more parameters than the smaller MLP. Training the large model on the proxy-selected sequence is still significantly faster than the uniform sampling baseline. This can be explained by the high rank correlation of the loss between the proxy model and the large model: the models rank points similarly, providing further validation for the use of a proxy model. As a result, we can save significant computation.

Table 1. Size and architecture of the proxy and large model used in QMNIST experiments.

Model	Num of Params.	FLOPS
3-layer MLP with 128 hidden units	135 k	136 k
3-layer MLP with 512 hidden units	932 k	935 k

On CIFAR and CINIC, we go one step further by using a proxy model which is not only smaller but also a significantly different and far simpler architecture than the large model. We use a small 1990s-style CNN, reminiscent of LeNet (LeCun et al., 1998), as our proxy model, while using

ResNet-18 (He et al., 2015) as our larger model. ResNet-18 uses approximately 29× more floating point operations per second (FLOPS) than the small CNN. This is in addition to using a very different architecture, with residual connections amongst other differences.

Table 2. Size and architecture of the proxy and large model used in CIFAR10 and CINIC10 experiments.

Model	Num of Params.	FLOPS
Small CNN	0.538 M	0.019 G
ResNet-18	11.17 M	0.557 G

Despite the significant differences, we show that using the sequence obtained using the reducible loss acquisition function with a proxy model, we can speed up training, and thus reduce computational costs. Hence, we can use the selected sequence to train other models. Storing the sequence is cheap since we only save the indices i of each x_i. This lends itself well to a variety of settings in which the corpus is large, and computational costs are high which is often the case in low-resource environments. In future work, this should also be extended to large-scale applications such as language modelling and contrastive learning, to reduce their training speed and computation.

References

Anonymous. Active learning under pool set distribution shift and noisy data. *ICML Workshop Submission*, 2021.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Brinker, K. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 59–66, 2003.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Darlow, L. N., Crowley, E. J., Antoniou, A., and Storkey, A. J. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

Der Kiureghian, A. and Ditlevsen, O. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Farquhar, S., Gal, Y., and Rainforth, T. On statistical bias in active learning: How and when to fix it. *arXiv preprint arXiv:2101.11665*, 2021.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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

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

Kirsch, A., Van Amersfoort, J., and Gal, Y. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *arXiv preprint arXiv:1906.08158*, 2019.

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

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

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

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Thulasidasan, S., Bhattacharya, T., Bilmes, J., Chennupati, G., and Mohd-Yusof, J. Combating label noise in deep learning using abstention. *arXiv preprint arXiv:1905.10964*, 2019.

Tian, Y., Henaff, O. J., and Oord, A. v. d. Divide and contrast: Self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*, 2021.

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