REDUCR: ROBUST DATA DOWNSAMPLING USING CLASS PRIORITY REWEIGHTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern machine learning models are becoming increasingly expensive to train for real-world image and text classification tasks, where massive web-scale data is collected in a streaming fashion. To reduce the training cost, online batch selection techniques have been developed to choose the most informative datapoints. However, these techniques can suffer from poor worst-class generalization performance due to class imbalance and distributional shifts. This work introduces REDUCR, a robust and efficient data downsampling method that uses class priority reweighting. REDUCR *reduces* the training data while preserving worst-class generalization performance. REDUCR assigns priority weights to datapoints in a class-aware manner using an online learning algorithm. We demonstrate the data efficiency and robust performance of REDUCR on vision and text classification tasks. On web-scraped datasets with imbalanced class distributions, REDUCR significantly improves worst-class test accuracy (and average accuracy), surpassing state-of-the-art methods by around 15%.

1 INTRODUCTION

The abundance of data has had a profound impact on machine learning (ML), both positive and negative. On the one hand, it has enabled ML models to achieve unprecedented performance on a wide range of tasks, such as image and text classification (Kuznetsova et al., 2020; He et al., 2015; Brown et al., 2020; Tran et al., 2022; Anil et al., 2023). On the other hand, training models on such large datasets can be computationally expensive and time-consuming (Kaddour et al., 2023), making it unsustainable in some situations (Bender et al., 2021; Patterson et al., 2021). Additionally, the high speed at which streaming data is collected can make it infeasible to train on all of the data before deployment. To tackle these issues, various methods have emerged to selectively choose training data, either through pre-training data pruning (Sorscher et al., 2022; Bachem et al., 2017) or online batch selection techniques (Loshchilov & Hutter, 2016; Mindermann et al., 2022), ultimately reducing data requirements, improving training efficiency, and enabling ML models to handle otherwise unmanageable large and complex datasets.

However, in real-world settings, a variety of factors can affect the selection of datapoints, such as noise (Xiao et al., 2015; Cao et al., 2021; Wei et al., 2021) and class-imbalance in the data (Van Horn et al., 2018; Philip & Chan, 1998; Radivojac et al., 2004). Online selection methods can exacerbate these problems by further reducing the number of datapoints from underrepresented classes, which can degrade the performance of the model on those classes (Buda et al., 2018; Cui et al., 2019). Moreover, distributional shift (Koh et al., 2021) between training and test time can lead to increased generalization error if classes with poor generalization error are overrepresented at test time.

In this work, we introduce REDUCR, which is a new online batch selection method that is *robust* to noise, imbalance, and distributional shifts. REDUCR employs multiplicative weights update to reweight and prioritize classes that are performing poorly during online batch selection. Figure 1 illustrates the intuition on how the method works. REDUCR can effectively reduce the training data and improve training efficiency while preserving the worst-class generalization performance of the model. For example, on the Clothing1M dataset (Xiao et al., 2015), Figure 2 shows that, compared to the best performing online batch selection methods, REDUCR achieves around a 15% boost in performance for the worst-class test accuracy.

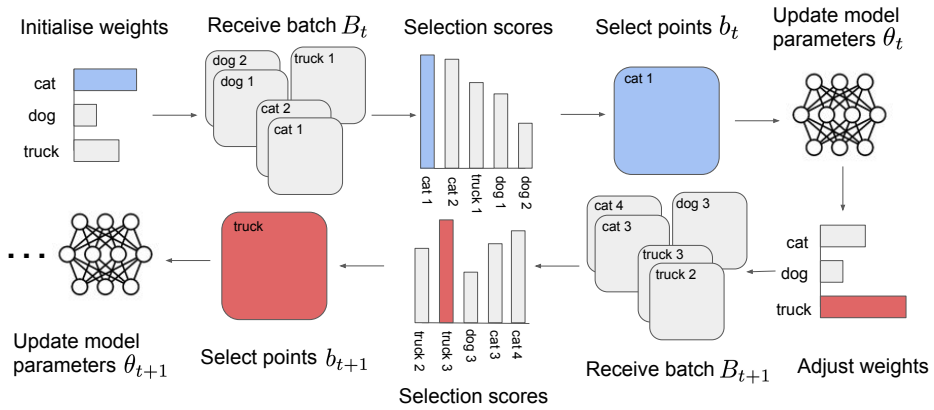


Figure 1: REDUCR starts by initializing weights of classes. At each timestep t, the model receives a batch of datapoints B_t. REDUCR computes the selection scores for each datapoint based on its usefulness to the model and the class weights, and selects new datapoints $b_t \subset B_t$ that achieve the highest selection scores. After the model takes gradient steps on the selected datapoints, REDUCR adjusts the weights to reflect increased priorities on underperforming classes.

Main contributions. (1) We formalise the maximin problem of robust data selection (§3). (2) We propose the REDUCR algorithm, which is equipped with a new robust selection rule that evaluates how much a datapoint will affect the generalization error of a specific class (§4.2). (3) We evaluate our algorithm on a series of text and image classification tasks and show that it achieves strong worst-class test accuracy while frequently surpassing state-of-the-art methods in terms of average test accuracy(§5).

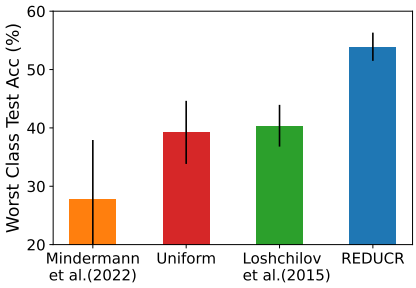


Figure 2: REDUCR significantly improves worst-class test accuracy on Clothing1M.

Related work. Mindermann et al. (2022) have developed an online batch selection method called RHO-LOSS, which uses a *reference model* trained on a holdout dataset to guide the selection of points during training. Certain extensions of this work have focused on using a reference model in different settings such as reinforcement learning (Sujit et al., 2022). However, to our knowledge, none have focused on improving the worst-class generalisation performance. Other batch selection methods (Loshchilov & Hutter, 2016; Jiang et al., 2019; Kawaguchi & Lu, 2020) use the training loss of points under the model or an approximate gradient norm (Katharopoulos & Fleuret, 2017) to select challenging points. We observe that these methods (e.g., see Loshchilov & Hutter (2016) in Figure 2) exhibit greater consistency in terms of worst-class generalization error in imbalanced datasets. Nevertheless, Loshchilov & Hutter (2016) do not surpass the average generalization error achieved by point selection with a reference model, namely, RHO-LOSS.

Recently, several works have also used reference models or a holdout dataset to train robust models. Oren et al. (2019); Liu et al. (2021); Clark et al. (2019) use a reference model to identify difficult-to-learn groups (or points, or biases) during training. Han et al. (2018) use two models which act as a reference model for the other to remove noisy points from the training data. Cao et al. (2021); Ren et al. (2018) use a holdout dataset to reweight points or their regularization during training to achieve the best loss on the validation holdout dataset. Sagawa et al. (2020) reweight groups known at training time and focus on fighting spurious correlations and improving worst-group generalisation error. In contrast, in our setting, class labels are available and we measure the performance in terms of worst-class generalisation error. Moreover, whilst these works aim to train robust models they do not consider efficient data selection strategies.

Xie et al. (2023) use both weights update rules and a reference model to find mixtures of corpora in LLM pretraining resulting in improved performance and training speed. Besides the problem setup, our method differs in three ways: i) we focus upon online batch selection; ii) we use multiple reference models; iii) and we use a class-holdout loss term (see Equation (7)) to reweight batches.

Finally, efficient data selection is a well-explored problem with various approaches, including active learning methods when label information is unknown (MacKay, 1992; Houlsby et al., 2011; Kirsch et al., 2019; 2023; Ash et al., 2020); data pruning and coreset techniques for pre-training data selection (Sorscher et al., 2022; Bachem et al., 2017; Borsos et al., 2020; Coleman et al., 2020); data

distillation approaches (Cazenavette et al., 2022; Nguyen et al., 2021); and non-parametric inducing point methods (Galy-Fajou & Opper, 2021).

2 Background

We consider a C-way classification task and denote a model as $p(y \mid x, \theta)$, where x denotes an input and $y \in [C]$ the corresponding class label; the model is parameterized by θ. For any training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ with N datapoints, we use a point estimate of θ to approximate the posterior model as $p(y \mid x, \mathcal{D}) \approx p(y \mid x, \hat{\theta})$. This estimate $\hat{\theta}$ can be obtained by running stochastic gradient descent (SGD) to optimize the cross-entropy loss over training dataset $\mathcal{D}$.

The goal of *data downsampling* is to select a dataset $\mathcal{D}_T \subset \mathcal{D}$ of size T ($\ll N$) for training such that the generalisation error of the resulting model is minimised. We write this objective in terms of a separate *holdout* dataset $\mathcal{D}_{ho} = \{(x_{ho,i}, y_{ho,i})\}_{i=1}^{N_{ho}}$ as follows:

$$\mathcal{D}_T = \underset{D \subset \mathcal{D}, |D|=T}{\operatorname{argmax}} \log p(\mathbf{y}_{ho} | \mathbf{x}_{ho}, D), \tag{1}$$

where the inputs and their labels are $\mathbf{x}_{ho} = [x_{i,ho}]_{i=1}^{N_{ho}}$ and $\mathbf{y}_{ho} = [y_{i,ho}]_{i=1}^{N_{ho}}$, respectively. Here, the likelihood of the holdout dataset is used as a proxy for the generalisation error. The problem is computationally prohibitive due to its combinatorial nature. Moreover, for a massive (or streaming) training dataset $\mathcal{D}$, it is not computationally possible to load $\mathcal{D}$ all at once and it is common to loop through the data by iteratively loading subsets.

Online batch selection is a practical streaming setup to approximate the data downsampling problem, where at each timestep t, the model observes a training data subset $B_t \subset \mathcal{D}$, and the goal is to iteratively select a small batch $b_t \subset B_t$ for the model to take gradient steps. A standard solution to this problem is to design a selection score function that take into account the labels of the data. The selection score function can then be used to *score* the utility of the small batch b_t. See Algorithm 2 in Appendix A.1 for an example method.

Reducible Holdout Loss (RHO-Loss) (Mindermann et al., 2022) is an online batch selection method that uses the performance on a holdout dataset as the selection scores for small batches. More precisely, for each timestep t, RHO-Loss selects

$$b_t = \underset{b \subset B_t}{\operatorname{argmax}} \log p(\mathbf{y}_{ho} \mid \mathbf{x}_{ho}, \mathcal{D}_t \cup b), \tag{2}$$

where $\mathcal{D}_t = \bigcup_{\tau=1}^{t-1} b_\tau$ is the cumulative training data the model has encountered until iteration t.

3 Problem Formulation

In this work, we introduce the *robust* data downsampling problem, where the goal is to select a training dataset $\mathcal{D}_T$ of size T such that worst-class performance is optimized. Let the holdout dataset with class $c \in [C]$ be $D_{ho}^{(c)} = \{(x, y) \in D_{ho} \mid y \equiv c\} = \{(x_{ho,i}^{(c)}, y_{ho,i}^{(c)})\}_{i=1}^{N_{ho}^{(c)}}$. We can write the objective of robust data downsampling as

$$\mathcal{D}_T = \underset{D \subset \mathcal{D}, |D|=T}{\operatorname{argmax}} \min_{c \in [C]} \log p(\mathbf{y}_{ho}^{(c)} \mid \mathbf{x}_{ho}^{(c)}, D), \tag{3}$$

where $\mathbf{x}_{ho}^{(c)} = [x_{ho,i}^{(c)}]_{i=1}^{N_{ho}^{(c)}}$ and $\mathbf{y}_{ho}^{(c)} = [y_{ho,i}^{(c)}]_{i=1}^{N_{ho}^{(c)}}$ correspond to the collections of inputs and labels in the class-specific holdout dataset $D_{ho}^{(c)}$.

Compared to Equation (1), the objective in Equation (3) is even more challenging because of the maximin optimisation that involves C discrete classes. In fact, solving Equation (3) is known to be NP-hard, even when the objectives (each $p(\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \cdot)$, $c \in [C]$) are *submodular* set functions. Chen et al. (2017) demonstrate the application of zero-sum game no-regret dynamics, where a learner employs a $(1 - 1/e)$-near-optimal greedy strategy and an adversary seeks to find a distribution over loss functions that maximizes the learner's loss. In this scenario, a single set is identified, which, although larger than size T, achieves a constant-factor approximation.

Robust online batch selection approximates the robust data downsampling problem by taking into account the practical limitations of data operation. Namely, we assume a streaming setting where the model observes training data subset $B_t \subset \mathcal{D}$ at each timestep t. The goal is to select a small batch $b_t \subset B_t$ to compute gradients for model training with SGD, such that the model obtains top performance for the worst-class (Equation (3)). The robust setting motivates the development of novel batch selection methods that consider how each datapoint affects the generalization error on the worst-case class of inputs, rather than just the overall generalization error. Next, we introduce a new selection rule that achieves this and propose a practical algorithm for its implementation.

4 REDUCR FOR ROBUST ONLINE BATCH SELECTION

We propose REDUCR, a robust and efficient data downsampling method using class priority reweighting to solve the robust online batch selection problem in Section 3. The batch selection strategy of REDUCR relates the effect of training on a batch of candidate points b_t to the generalization error of a specific class in the holdout dataset.

4.1 ONLINE LEARNING

To solve Equation (3) in an online manner, we propose to use class priority reweighting, a variant of the multiplicative weights update method (Freund & Schapire, 1997; Cesa-Bianchi & Lugosi, 2006). At the beginning of training we initialise a weight vector $\mathbf{w}_0$ over a C dimensional simplex, $\Delta = \{\mathbf{w} = [w_c]_{c=1}^C \in \mathbb{R}^C | \sum_{c=1}^C w_c = 1\}$. Each element of $\mathbf{w}_0$ is initialised to be $w_{0,c} = 1/C$. For each iteration t, small batch $b_t \subset B_t$ is chosen by maximising the weighted sum of the C different class-specific scoring functions (i.e., by best-responding to the current class-weights $\mathbf{w}_t$),

$$b_t = \underset{b \subset B_t}{\operatorname{argmax}} \sum_{c=1}^{C} w_{t,c} \left(\log p(\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t \cup b) \right), \tag{4}$$

where $\mathcal{D}_t = \bigcup_{\tau=1}^{t-1} b_\tau$, $\mathbf{w}_t = [w_{t,c}]_{c=1}^C \in \Delta$, and

$$w_{t,c} = w_{t-1,c} \frac{\exp\left(-\eta \log p(\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t)\right)}{\sum_{j=1}^C w_{t-1,j} \exp\left(-\eta \log p(\mathbf{y}_{ho}^{(j)} | \mathbf{x}_{ho}^{(j)}, \mathcal{D}_t)\right)}. \tag{5}$$

In the previous alternating procedure, class-weights are updated multiplicatively according to how well they perform given the selected batch (they increase for poorly performing classes and decrease otherwise). In Equation (5), η is a learning rate that adjusts how concentrated the probability mass is in the resulting distribution. Figure 1 shows an intuitive illustration of how reweighting works in practice where classes that perform badly have low data likelihoods and are thus upweighted by Equation (5). We next introduce how to compute the likelihoods for class-specific holdout sets, i.e., $p(\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t \cup b)$ in Equation (4).

4.2 COMPUTING SELECTION SCORES

Given the current dataset $\mathcal{D}_t$ at timestep t and additional datapoints $b \subset B_t$, we would like to compute the likelihood of the holdout dataset that belongs to class c. For simplicity, we consider the case where the small batch to be selected only includes a single datapoint, i.e., $b = \{(x, y)\}$. We express the objective using the Bayesian perspective introduced in Section 2, i.e.,

$$\begin{aligned} \log p(\mathbf{y}_{ho}^{(c)} \,|\, \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t \cup \{(x,y)\}) &= \log \frac{p(y|x, \mathcal{D}_{ho}^{(c)}, \mathcal{D}_t) p(\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, x, \mathcal{D}_t)}{p(y|x, \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t)} = \log \frac{p(y|x, \mathcal{D}_{ho}^{(c)}, \mathcal{D}_t) p(\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t)}{p(y|x, \mathcal{D}_t)} \\ &= -\log p(y \,|\, x, \mathcal{D}_t) + \log p(y \,|\, x, \mathcal{D}_t, \mathcal{D}_{ho}^{(c)}) + \log p(\mathbf{y}_{ho}^{(c)} \,|\, \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t). \end{aligned} \tag{6}$$

Equation (6) follows from the application of the Bayes rule and the conditional independence of x and $\mathbf{x}_{ho}^{(c)}$ with $\mathbf{y}_{ho}^{(c)}$ and y, respectively. The posterior terms in Equation (6) can be approximated with point estimates of model parameters (see §2). Computing Equation (6) involves two models: (1) the *target* model with parameters θ_t, which is trained on the cumulative training dataset $\mathcal{D}_t = \bigcup_{\tau=1}^{t-1} b_\tau$; (2) a *class-irreducible loss model* (following the terminology from Mindermann et al. (2022)) with parameters $\theta_t^{(c)}$, which is trained on $\mathcal{D}_t$ and class-specific holdout data $\mathcal{D}_{ho}^{(c)}$. The target model is what we are interested in for the classification task.

We use $\mathcal{L}[y|x,\theta] = -\log p(y \mid x, \theta)$ to denote the cross-entropy loss for any model parameters θ, and we re-write Equation (6) as follows,

$$\log p(\mathbf{y}_{ho}^{(c)} \mid \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t \cup \{(x,y)\}) \approx \underbrace{\mathcal{L}[y|x,\theta_t]}_{\text{model loss}} - \underbrace{\mathcal{L}[y|x,\theta_t^{(c)}]}_{\text{class-irreducible loss}} - \underbrace{\mathcal{L}[\mathbf{y}_{ho}^{(c)}|\mathbf{x}_{ho}^{(c)},\theta_t]}_{\text{class-holdout loss}}. \tag{7}$$

We name the three terms in Equation (7) the *model loss*, *class-irreducible loss* and *class-holdout loss*, respectively. We define the term *excess loss* as the difference of the model loss and class-irreducible loss. The excess loss is the improvement in loss for point (x, y) by observing more data from class c (i.e., $\mathcal{D}_{ho}^{(c)}$). The class-irreducible loss and the class-holdout loss both depend on the class c.

Intuitively, if two data points are from different classes, REDUCR will take into account the weight of the worst-performing class, which is reflected by the class-holdout loss. This ensures that REDUCR is focusing on improving the performance of the model on the classes that are most difficult to learn. In a different scenario, if two datapoints are from the same class, their class-holdout losses will be the same, and the point with a larger excess loss will be preferred. This means that REDUCR prefers datapoints whose losses have more potential to be improved.

Computing the approximate in Equation (7) is far more tractable than naively re-training a new model (i.e., $\log p(\mathbf{y}_{ho}^{(c)}|\mathbf{x}_{ho}^{(c)}, \mathcal{D}_t \cup \{(x,y)\})$) for each possible candidate point (x, y). The model loss and the class-holdout loss only require evaluating the cross-entropy losses of some datapoints on the target model. More generaly, if batch b can include more than one point, we can simply change the x and y to a list of inputs and labels instead. Next, we further improve the efficiency of REDUCR by approximating the class-irreducible loss model.

4.3 Class-Irreducible Loss Models

For each selected batch b_t under the current selection rule in Equation (7), we need to update C class-irreducible loss models to compute the class-irreducible losses. We propose to approximate these models using *amortised* class-irreducible loss models, which are trained for each class at the beginning of REDUCR and do not need to be updated at future timesteps.

We interpret the class irreducible loss term as an expert model at predicting the label of points from a specific class c due to the extra data from the holdout dataset this term has available. To create an approximation of this expert model, we train the amortised class-irreducible loss models using an adjusted loss function in which points with a label from the class c are up-weighted by a parameter $\gamma \in (0, +\infty)$ (set in Section 5):

$$\phi_c = \arg\min_{\phi} \sum_{(x,y)\in\mathcal{D}_{ho}} (1 + \gamma\, \mathbb{I}[c \equiv y])\, \mathcal{L}[y|x,\phi]. \tag{8}$$

Here we define $\mathbb{I}[\cdot]$ as the indicator function. Equation (8) optimizes over the parameters of the amortised class-irreducible loss model for class c, and obtain ϕ_c to approximate $\theta_t^{(c)}$ in Equation (7), i.e., $\mathcal{L}[y|x,\theta_t^{(c)}] \approx \mathcal{L}[y|x,\phi_c]$. Algorithm 3 details the full amortised class-irreducible loss model training procedure in Appendix A.2. We further motivate our approximation in Appendix A.3.

4.4 REDUCR as a practical algorithm

We use the selection objective in Equation (7) along with the amortised class-irreducible loss model approximation (Section 4.3) and the online algorithm (Section 4.1) to reweight the worst performing class during training and select points that improve its performance. See Algorithm 1 for a full description of the REDUCR method. At each iteration, the top k points are selected (Line 6) according to the weighted sum of Equation (7) for each class $c \in C$, thus efficiently approximating the combinatorial problem from Equation (4). As the class-holdout loss does not depend on the selected points b_t and we sum over the classes, we can remove this term from the weighted sum of the selection scores and only apply it when updating the weights $\mathbf{w}_t$ (in Line 7 and 8). We calculate the *average* class-holdout loss to remove any dependence of the term upon the size of the classes in the holdout dataset. We find that clipping the excess loss improves the stability of the algorithm in practice. We test this heuristic empirically in Section 5.2 and provide an intuitive explanation for why this is the case in Appendix A.7.2.

Algorithm 1 REDUCR for robust online batch selection

1: **Input:** data pool $\mathcal{D}$, holdout data $\mathcal{D}_{ho} = \bigcup_{c \in C} \mathcal{D}_{ho}^{(c)}$, learning rate $\eta \in (0, \infty)$, small batch size k, total timesteps T/k
2: Initialize class weights $\mathbf{w}_1 = \frac{1}{C}\mathbf{1}_C$
3: Use $\mathcal{D}_{ho}$ to train C amortised class irreducible loss models as per Equation (8) to obtain ϕ_c
4: **for** $t \in [T/k]$ **do**
5: Receive batch $B_t \subset \mathcal{D}$
6: $b_t = \underset{b \subset B_t : |b| = k}{\operatorname{argmax}} \sum_{(x,y) \in b} \sum_{c \in C} w_{t,c} \max\left(0, \mathcal{L}[y|x, \theta_t] - \mathcal{L}[y|x, \phi_c]\right)$ ▷ Select points with top k selection scores
7: Compute the objective value for every class $c \in C$:

$$\alpha_c \leftarrow \sum_{(x,y) \in b_t} \left(\max(0, \mathcal{L}[y|x, \theta_t] - \mathcal{L}[y|x, \phi_c]) - \mathcal{L}[\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \theta_t] \right)$$

8: Update class weights for every class $c \in C$: $w_{t+1,c} = w_{t,c} \frac{\exp(-\eta \alpha_c)}{\sum_{j \in C} w_{t,j} \exp(-\eta \alpha_j)}$
9: $\theta_{t+1} \leftarrow SGD(\theta_t, b_t)$
10: **end for**

When comparing REDUCR to other online batch selection methods, we observe distinct batch selection patterns. When the dataset is class-imbalanced, the underrepresented classes tend to perform worse because of the lack of training data from those classes. RHO-LOSS may struggle to select points from the underrepresented classes as they have less effect on the loss of the holdout dataset. Selection rules that select points with high training loss (Loshchilov & Hutter, 2016; Kawaguchi & Lu, 2020; Jiang et al., 2019) might select points from the underrepresented classes but have no reference model to determine which of these points are learnable given more data and thus noisy or task-irrelevant points may be selected. In contrast, REDUCR addresses both of these issues by identifying underrepresented classes and using the class-irreducible loss model to help to determine which points from these classes should be selected.

Even when the dataset is not imbalanced, certain classes might be difficult to learn; for example, due to noise sources in the data collection processes. Via Equation (5), REDUCR is able to reweight the selection scores such that points that are harder to learn from worse-performing classes are selected over points that are easier to learn from classes that are already performing well. This is in contrast to RHO-LOSS which will always select points that are easier to learn. We empirically demonstrate this on class balanced datasets in Section 5.

5 EXPERIMENTS

In this section, we present empirical results to showcase the performance of REDUCR on large-scale vision and text classification tasks.

Datasets. We train and test REDUCR on image and text datasets. We use CIFAR10 (Krizhevsky et al., 2009), CINIC10 (Darlow et al., 2018), Clothing1M (Xiao et al., 2015), the Multi-Genre Natural Language Interface (MNLI), and the Quora Question Pairs (QQP) datasets from the GLUE NLP benchmark (Wang et al., 2019). Each dataset is split into a labelled training, validation and test dataset (for details see Appendix A.5), the validation dataset is used to train the class-irreducible loss models and evaluate the class-holdout loss during training. The Clothing1M dataset uses 100k additional points from the training dataset along with the validation dataset to train the irreducible loss model(s) (as per (Mindermann et al., 2022)). We simulate the streaming setting by randomly sampling batch B_t from dataset $\mathcal{D}$ at each timestep.

Models. For the experiments on image datasets (CIFAR10, CINIC10 and Clothing1M) all models use a ResNet-18 model architecture (He et al., 2016). For the Clothing1M dataset we use a ResNet-18 model pretrained on the imagenet dataset (Deng et al., 2009). The networks are optimised with AdamW (Loshchilov & Hutter, 2019) and the default Pytorch hyperparameters are used for all methods except CINIC10 for which the weight decay is set to a value of 0.1. For the NLP dataset we use the *bert-base-uncased* (Devlin et al., 2019) model from HuggingFace (Wolf et al., 2020) and set the optimizer learning rate to $1e^{-6}$.

Dataset	Worst-Class Test Accuracy (%) ± 1 std			
	UNIFORM	TRAIN LOSS	RHO-LOSS	REDUCR
CIFAR10 (10 runs)	75.01 ± 1.37	76.1 ± 2.31	78.80 ± 2.09	**83.29 ± 0.84**
CINIC10 (10 runs)	64.70 ± 2.45	64.83 ± 4.75	69.39 ± 3.56	**75.30 ± 0.85**
Clothing1M (5 runs)	39.23 ± 5.41	40.37 ± 3.58	27.77 ± 10.16	**53.91 ± 2.42**
MNLI (5 runs)	74.70 ± 1.26	74.56 ± 1.44	76.74 ± 0.93	**79.45 ± 0.39**
QQP (5 runs)	73.21 ± 2.04	79.96 ± 2.34	78.21 ± 1.95	**86.61 ± 0.49**

Table 1: REDUCR outperforms RHO-LOSS (the best overall baseline) in terms of the worst-class test accuracy on Clothing1M, CINIC10 and CIFAR10 by at least 5-26%. Across all baselines, REDUCR gains about 15% more accuracy on the noisy and imbalanced Clothing1M dataset as shown in Figure 2.

Dataset	Average Test Accuracy (%) ± 1 std			
	UNIFORM	TRAIN LOSS	RHO-LOSS	REDUCR
CIFAR10 (10 runs)	85.09 ± 0.52	88.86 ± 0.22	**90.00 ± 0.33**	**90.02 ± 0.44**
CINIC10 (10 runs)	79.51 ± 0.30	79.25 ± 0.33	**82.09 ± 0.30**	**81.68 ± 0.47**
Clothing1M (5 runs)	69.60 ± 0.85	69.63 ± 0.30	71.07 ± 0.46	**72.69 ± 0.42**
MNLI (5 runs)	79.19 ± 0.53	76.85 ± 0.14	**80.89 ± 0.31**	**80.28 ± 0.33**
QQP (5 runs)	85.05 ± 0.43	**86.30 ± 0.41**	**86.88 ± 0.31**	**86.99 ± 0.49**

Table 2: REDUCR matches or outperforms the average test accuracy of the best competing baseline across all datasets. Note that optimizing the average test accuracy is not the objective of REDUCR. These results, together with Table 1, demonstrate the significant advantage of REDUCR to improve the worst-class accuracy while maintaining the strong average-case performance.

Baselines. We benchmark our method against the state-of-the-art RHO-LOSS (Mindermann et al., 2022) and Loshchilov & Hutter (2016), an online batch selection method that uses the training loss to select points. We refer to the latter baseline as TRAIN LOSS. We also compare against UNIFORM where points are chosen at random from the large batch at each training step.[1] All experiments are run multiple times and the mean and standard deviation across runs calculated. Unless stated otherwise 10% of batch B_t is selected as the small batch b_t, and we set $\eta = 1e-4$. $\gamma = 9$ is used when training each of the amortised class-irreducible loss models on the vision datasets and $\gamma = 4$ for the NLP datasets. We study the impact of γ and η on REDUCR further in Appendix A.9. For full details of the experimental setup see Appendix A.5.[2]

5.1 KEY RESULTS

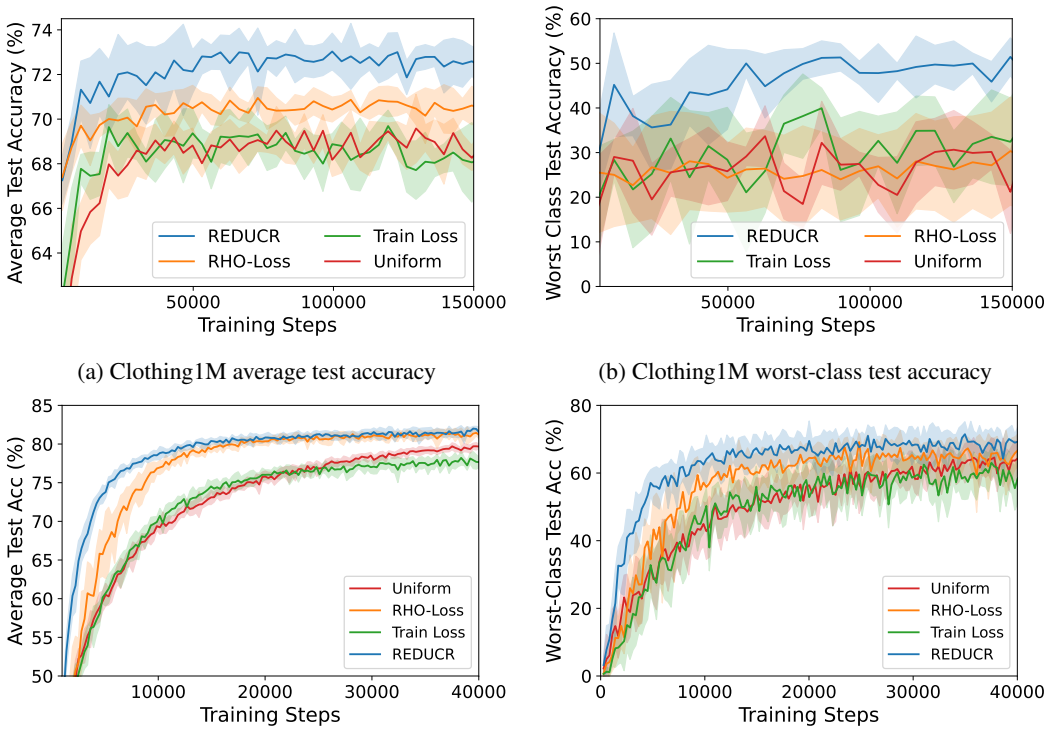
The worst-class and average test accuracy for the datasets and model are shown in Table 1 and Table 2, respectively. Across all datasets, REDUCR outperforms the baselines in terms of the worst-class accuracy and matches or even outperforms the average test accuracy of RHO-LOSS within one standard deviation. This is also surprising because the primary goal of REDUCR is not to optimize the overall average (over classes) performance.

REDUCR performs particularly strongly on the Clothing1M dataset: Table 1 shows REDUCR improves the worst-class test accuracy by around 15% when compared to TRAIN LOSS, the next best-performing baseline, and by around 26% when compared to RHO-LOSS, the overall best-performing baseline across datasets. Figure 3b shows that REDUCR also achieves this performance in a more data efficient manner than the comparable baselines, achieving a mean worst-class test accuracy of 40% within the first 10k training steps. We also observe improved efficiency on the CINIC10 dataset as shown in Figure 3c and Figure 3d and the MNLI and QQP datasets as detailed in Figure 6.

The Clothing1M dataset also sees a distribution shift between the training and test dataset. In the test dataset, the worst performing class is much more prevalent than in the training dataset and as such improvements to its performance impact the average test accuracy significantly. Figure 3a shows the

[1] We use training step and timestep interchangeably.

[2] Code available at: https://anonymous.4open.science/r/REDUCR-24D3



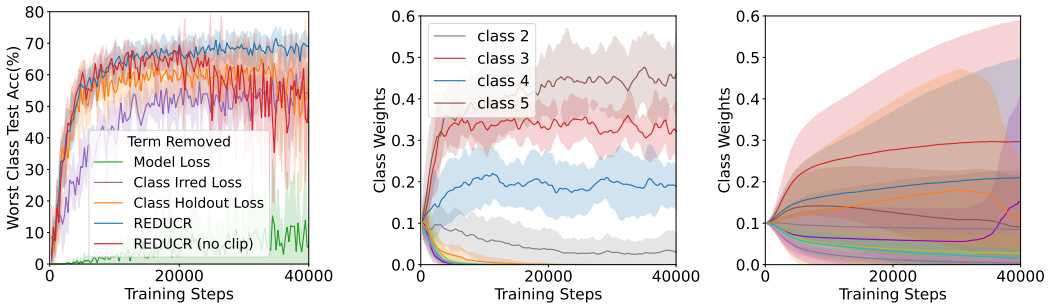
(a) Clothing1M average test accuracy

(b) Clothing1M worst-class test accuracy

(c) CINIC10 average test accuracy

(d) CINIC10 worst-class test accuracy

Figure 3: REDUCR improves the a) average and b) worst-class test accuracy on the Clothing1M dataset when compared with the RHO-LOSS, TRAIN LOSS and UNIFORM baselines. On CINIC10 it matches the performance of the best baseline for c) the average test accuracy, and outperforms the best baseline in d) the worst-class test accuracy, whilst improving the data efficiency in both.



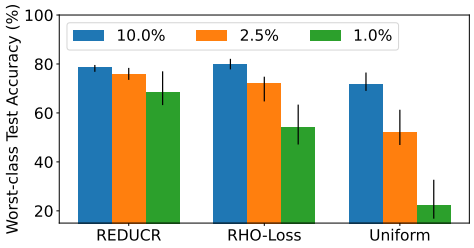
(a) (b) (c)

Figure 4: a) The worst-class test accuracy *decreases* when the model loss, class irreducible loss, and class-holdout loss terms are removed from REDUCR on CINIC10. Comparing REDUCR with clipping for excess losses (Algorithm 1) and REDUCR (no clip) which removes the clipping, we observe that REDUCR achieves more stable performance. We show the class weights $\mathbf{w}$ at each training step for b) REDUCR and c) REDUCR with the class-holdout loss term ablated. The ablation model fails to consistently prioritise the underperforming classes across multiple runs.

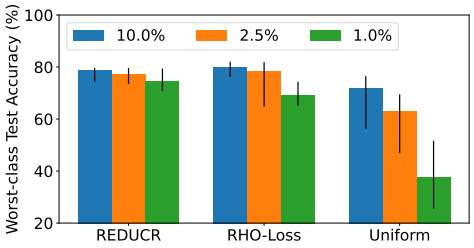
impact of this distribution shift as the improved performance of the model on the worst-class results in an improved average test accuracy to the state-of-the-art RHO-LOSS baseline.

5.2 ABLATION STUDIES

To further motivate the selection rule in Equation (7), we conduct a series of ablation studies to show that all the terms are necessary for robust online batch selection. Figure 4a shows the performance



(a) Under-sampling on class 3



(b) Under-sampling on class 5

Figure 5: Results on worst-class test accuracy for artificially imbalancing (1) class 3 and (b) class 5 on CIFAR10, with 1%, 2.5%, 10.0% percent imbalances (percentage of training data that belong to the imbalanced class). REDUCR significantly reduces the deterioration in performance, showing that target models trained using our method are more robust to class imbalance than models trained using the RHO-LOSS and UNIFORM baselines.

of REDUCR on the CINIC10 dataset when the model loss, amortised class-irreducible loss and class-holdout loss terms of the algorithm were individually excluded from the selection rule. We note that all three terms in Equation (7) are required to achieve a strong worst-class test accuracy.

The removal of the class-holdout loss term affects the ability of REDUCR to prioritise the weights of the model correctly. In Figure 4 we compare the class weights of REDUCR and an ablation model without the class-holdout loss term. The standard model clearly prioritises classes 3, 4 and 5 during training across all 5 runs, whilst the ablation model does not consistently weight the same classes across multiple runs. We also conducted an ablation study on the clipping of the excess loss to motivate its inclusion in the algorithm, this is also shown in Figure 4a, we note that this stabilises the model performance towards the end of training and investigate further in Appendix A.7.2.

5.3 IMBALANCED DATASETS

In this section, we investigate the performance of models trained using REDUCR on imbalanced datasets. We artificially imbalance the CIFAR10 dataset such that a datapoint of the imbalanced class is sampled with probability $p \in (0, 1/C]$ (referred to as the percent imbalance) and datapoints from the remaining classes are sampled with probability $(1-p)/(C-1)$ during model training. We only artificially imbalance the training and validation sets and not the test set. We conduct experiments with 1.0%, 2.5% and 10.0% percent imbalances on classes 3 and 5. Note that a percent imbalance of 10.0% is equivalent to the original (balanced) CIFAR10 training and validation sets. We repeat the experiments 10 times and plot the median values in Figure 5, and the error bars denote the best accuracy and the worst across 10 runs.

We find that the performance of models trained using REDUCR deteriorates less than those trained with the RHO-LOSS or UNIFORM baselines as the percent imbalance of a particular class decreases (see Figure 5). For example, when class 3 is imbalanced, in the most imbalanced case (1.0%) the median performance of REDUCR outperforms that of RHO-LOSS run by 14%. This demonstrates the effectiveness of REDUCR in prioritising the selection of data points from underrepresented classes.

6 CONCLUSION AND FUTURE WORK

In summary, we identified the problem of class-robust data downsampling and proposed a new method, REDUCR, to solve this problem using class priority reweighting. Our experimental results indicate that REDUCR significantly enhances data efficiency during training, achieving superior test accuracy for the worst-performing class and frequently surpassing state-of-the-art methods in terms of average test accuracy. REDUCR excels in setting where the available data is imbalanced by prioritising the selection of points from underrepresented classes.

An important consideration of future work should be the further reduction of computational resources necessary to use REDUCR. Combining the C amortised class-irreducible loss models into a single architecture would reduce both memory and computational requirements of the method.

REPRODUCIBILITY STATEMENT

All the experiments mentioned in the Paper and Appendix were implemented using the code provided in `https://anonymous.4open.science/r/REDUCR-24D3` which includes the necessary code for processing the raw datasets, which are freely available online.

REFERENCES

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. PaLM 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *International Conference on Learning Representations*, 2020.

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 2018.

Kaidi Cao, Yining Chen, Junwei Lu, Nikos Arechiga, Adrien Gaidon, and Tengyu Ma. Heteroskedastic and imbalanced deep learning with adaptive regularization. *International Conference of Learning Representations*, 2021.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *International Conference of Computer Vision*, 2022.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Robert S Chen, Brendan Lucier, Yaron Singer, and Vasilis Syrgkanis. Robust optimization for non-convex objectives. *Advances in Neural Information Processing Systems*, 2017.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. *arXiv preprint arXiv:1909.03683*, 2019.

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.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *International Conference on Computer Vision*, 2019.

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.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *International Conference on Computer Vision*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997.

Théo Galy-Fajou and Manfred Opper. Adaptive inducing points selection for gaussian processes. *arXiv preprint arXiv:2107.10066*, 2021.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in Neural Information Processing Systems*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *International Conference on Computer Vision*, 2016.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

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.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023.

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

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

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in Neural Information Processing Systems*, 2019.

Andreas Kirsch, Sebastian Farquhar, Parmida Atighehchian, Andrew Jesson, Frederic Branchaud-Charron, and Yarin Gal. Stochastic batch acquisition for deep active learning. *Transactions on Machine Learning Research*, 2023.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 2020.

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*. PMLR, 2021.

Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *International Conference on Learning Representations, Workshop Track*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2019.

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 1992.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, 2010.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*. PMLR, 2022.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 2021.

Yonatan Oren, Shiori Sagawa, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust language modeling. *arXiv preprint arXiv:1909.02060*, 2019.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

K Philip and SJS Chan. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998.

Predrag Radivojac, Nitesh V Chawla, A Keith Dunker, and Zoran Obradovic. Classification and knowledge discovery in protein databases. *Journal of Biomedical Informatics*, 2004.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*. PMLR, 2018.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *Internation Conference on Learning Representations*, 2020.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 2000.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 2022.

Shivakanth Sujit, Somjit Nath, Pedro HM Braga, and Samira Ebrahimi Kahou. Prioritizing samples in reinforcement learning with reducible loss. *arXiv preprint arXiv:2208.10483*, 2022.

Dustin Tran, Jeremiah Liu, Michael W Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, et al. Plex: Towards reliability using pretrained large model extensions. *arXiv preprint arXiv:2207.07411*, 2022.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *International Conference on Computer Vision*, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *International Conference on Learning Representations*, 2019.

Tong Wei, Jiang-Xin Shi, Wei-Wei Tu, and Yu-Feng Li. Robust long-tailed learning under label noise. *arXiv preprint arXiv:2108.11569*, 2021.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *International Conference on Computer Vision*, 2015.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *arXiv preprint arXiv:2305.10429*, 2023.

A APPENDIX

A.1 ONLINE BATCH SELECTION PSEUDO-CODE

For the sake of convenience, we provide the pseudocode of the online batch selection protocol described in Section 2.

Algorithm 2 Online batch selection

Input: data pool $\mathcal{D}$, number of training steps T, stochastic gradient descent algorithm SGD, a loss function $\mathcal{L}$
for $t = 1$ to T **do**
  Sample batch B_t randomly from $\mathcal{D}$
  $b_t = \text{SelectBatch}(B_t, \theta_t)$
  L = $\sum_{(x_i,y_i)\in b_t} \mathcal{L}[y_i|x_i, \theta_t]$
  $\theta_{t+1} = \text{SGD}(L, \theta_t)$
end for

A.2 CLASS IRREDUCIBLE LOSS MODEL TRAINING PSEUDO-CODE

Here we detail the pseudo-code for training the class irreducible loss model described in Section 4.3

Algorithm 3 Class Reference Model Training

Input: holdout dataset $\mathcal{D}_{ho}$, number of training steps T, stochastic gradient descent algorithm SGD, a loss function $\mathcal{L}$, a specific class c
for $t = 1$ to T **do**
  $B_{ho} \sim \text{Uniform}(\mathcal{D}_{ho})$
  $L = \sum_{(x_i,y_i)\in B_{ho}} (1 + \gamma \mathbb{I}[c = y])\mathcal{L}[y_i|x_i, \phi_t]$
  $\phi_{t+1} = \text{SGD}(L, \phi_t)$
end for
return Class-irreducible loss model parameters ϕ_T

A.3 THE AMORTISED CLASS IRREDUCIBLE LOSS MODEL APPROXIMATION

The amortised class irreducible loss model is an important component in REDUCR as shown by our ablation study in Figure 4a. For each class $c \in [C]$, we approximate the second term of Equation (6), $\log p(y|x, \mathcal{D}_t, \mathcal{D}_{ho}^{(c)})$ via the model trained using Algorithm 3. This approximation has two steps: firstly we remove dependence of the class irreducible model loss on the training dataset at time t. A similar approximation is heavily explored by Mindermann et al. (2022) in Section 3 and Section 4 of their paper; in Appendix D of their work they show that this approximation is important for the stable training of RHO-LOSS. The approximation also aligns RHO-LOSS and REDUCR with other methods in the literature such as Xie et al. (2023); Oren et al. (2019) which similarly use a reference model that does not vary during training.

Secondly we up-weight data points in the loss function when their label $y \in [C]$ matches that of the specific class c. Unlike RHO-LOSS we cannot approximate the class irreducible loss as $\log p(y|x, \mathcal{D}_t, \mathcal{D}_{ho}^{(c)}) \approx \log p(y|x, \mathcal{D}_{ho}^{(c)})$ as this is a trivial model only trained on points with labels from a single class and thus does not provide a suitable signal to guide point selection.

We interpret the original class irreducible loss $\log p(y|x, \mathcal{D}_t, \mathcal{D}_{ho}^{(c)})$, as an expert model for class c as this model trains on extra points only sampled from that class, $\mathcal{D}_{ho}^{(c)}$. In our approximation we train on the holdout dataset which does not have excess examples of points from class c. We justify our up-weighting of points as a form of importance weighting (Shimodaira, 2000), where by up-weighting points with labels in a specific class we are calculating an approximation of the loss under a distribution in which points from class c are more prevalent.

A.4 The Effect of the Class-Holdout Loss on the Selection of Points

The class-holdout loss term only affects the selection of points at each iteration t through the selection of the weights $\mathbf{w}_t$. As it does not depend upon the candidate point $(x, y) \in B_t$ and the weights sum to one we can remove it from line 6 of Algorithm 1 and only include it in line 7 when we update the class weights. Similarly as the model loss does not depend upon the class c we can write the selection score as

$$\sum_{c \in C} w_{t,c} \log p(\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \mathcal{D}_t \cup (\{x, y\})) = \mathcal{L}[y|x, \theta_t] - \sum_{c \in C} w_{t,c}(\mathcal{L}[y|x, \theta_t^{(c)}]) - \sum_{c \in C} w_{t,c}(\mathcal{L}[\mathbf{y}_{ho}^{(c)} | \mathbf{x}_{ho}^{(c)}, \theta_t]). \tag{9}$$

A.5 Experiment Details

We provide the full code base anonymised for review purposes at: https://anonymous.4open.science/r/REDUCR-24D3.

CIFAR10 used half the training dataset (25k points) as a holdout validation dataset for training the amortised class-irreducible loss models and calculating the class-holdout loss during the robust online batch selection. We used the remaining 25k points as a training dataset and the provided test dataset (10k) for testing.

CINIC10 used the provided validation dataset for both the class holdout loss and amortised class irreducible loss models.

Clothing1M. The dataset consists of 1 million images labelled automatically using the keywords in its surrounding text. The dataset consists of 72k 'clean' images whose labels have been hand checked, 50k, 13k and 9k are respectively sorted into a clean training, validation, and test subdataset. To train the amortised class irreducible loss models we use 100k points randomly sampled from the union of the validation, clean and noisy training datasets. We calculate the class-holdout loss term and validation performance during training using the clean validation dataset.

MNLI. The dataset Williams et al. (2018) consists of 412k labeled sentence pairs; similarly to Sagawa et al. (2020) we split these sentence pairs into a train (206k), validation (164k), and labelled test (41k) dataset.

QQP. The dataset consists of 431k labeled sentence pairs; we remove points from class 1 to further imbalance the dataset resulting in 22% of the dataset labelled class 1. We split the remaining points into a train (148k), validation (67k), and labelled test (40k) dataset. We do not adjust the balance of the test dataset.

ResNet-18 used for the Clothing1M experiments is the pretrained model available via the Torchvision (Marcel & Rodriguez, 2010) model library. For the CIFAR10 and CINIC10 experiments we use the adapted ResNet-18 architecture detailed in Mindermann et al. (2022) Appendix B.

Train Loss baseline is taken from Loshchilov & Hutter (2016) where points from the large batch B_t are sampled with probability

$$p_i \propto \frac{1}{\exp(\log(s)/|B_t|)^i}. \tag{10}$$

Here p_i is the point with the ith highest training loss in the large batch. We set the selection pressure parameter $s_e = 100$ and do not vary this during training as per the Experiments in Section 6. of Loshchilov & Hutter (2016).

Compute. All models were trained on NVIDIA Tesla T4 GPUs.

Data Augmentation was applied to the training dataset during online batch selection and validation dataset during the training of the amortised class-irreducible loss model. We apply a random crop and random flip to the images.

A.6 Results with worst-class checkpointing

In Table 3 and Table 4 we show the worst-class and average test accuracy respectively, when the Uniform, Train Loss and RHO-Loss baselines use worst-class validation accuracy to check-

point the model during training. REDUCR still outperforms or matches the best baseline performance across all datasets. In the cases where REDUCR matches the performance of the best performing baseline, it does so in a more data efficient manner. Figure 6b and Figure 6d show the mean and standard deviation worst-class test accuracy across multiple runs on the QQP and MNLI datasets. REDUCR matches the best mean performance of the best performing baseline almost 100k training steps earlier on both datasets.

Dataset	**Worst-Class Test Accuracy (%) ± 1 std**			
	UNIFORM	**TRAIN LOSS**	**RHO-LOSS**	**REDUCR**
CIFAR10 (10 runs)	75.01 ± 1.37	79.32 ± 1.35	81.23 ± 1.18	$\mathbf{83.29 \pm 0.84}$
CINIC10 (10 runs)	70.86 ± 1.23	68.89 ± 0.86	$\mathbf{73.44 \pm 1.16}$	$\mathbf{75.30 \pm 0.85}$
Clothing1M (5 runs)	39.23 ± 5.41	49.02 ± 2.32	32.19 ± 9.83	$\mathbf{53.91 \pm 2.42}$
MNLI (5 runs)	76.88 ± 1.21	75.75 ± 0.56	$\mathbf{78.04 \pm 1.73}$	$\mathbf{79.45 \pm 0.39}$
QQP (5 runs)	84.50 ± 0.56	$\mathbf{85.49 \pm 1.32}$	82.60 ± 1.12	$\mathbf{86.61 \pm 0.49}$

Table 3: Worst-class test accuracy, when the RHO-LOSS and TRAIN LOSS baselines are checkpointed using their worst-class validation error during training.

Dataset	**Average Test Accuracy (%) ± 1 std**			
	UNIFORM	**TRAIN LOSS**	**RHO-LOSS**	**REDUCR**
CIFAR10 (10 runs)	85.09 ± 0.52	87.74 ± 0.50	$\mathbf{89.43 \pm 0.57}$	$\mathbf{90.02 \pm 0.44}$
CINIC10 (10 runs)	79.57 ± 0.75	78.21 ± 0.57	$\mathbf{81.28 \pm 0.54}$	$\mathbf{81.68 \pm 0.47}$
Clothing1M (5 runs)	69.60 ± 0.85	69.46 ± 0.43	70.63 ± 0.87	$\mathbf{72.69 \pm 0.42}$
MNLI (5 runs)	78.85 ± 0.38	78.50 ± 0.33	$\mathbf{80.50 \pm 0.45}$	$\mathbf{80.28 \pm 0.33}$
QQP (5 runs)	85.23 ± 0.36	$\mathbf{86.24 \pm 0.26}$	$\mathbf{86.75 \pm 0.37}$	$\mathbf{86.99 \pm 0.49}$

Table 4: Average test accuracy, when the RHO-LOSS and TRAIN LOSS baselines are checkpointed using their worst-class validation error during training.

A.7 ADDITIONAL EXPERIMENTAL RESULTS

In Appendix A.7.1 we show the per class weights for the Clothing1M dataset, whilst in Appendix A.7.2 we analyse the effect of the clipping term and provide some intuition behind its inclusion in the algorithm.

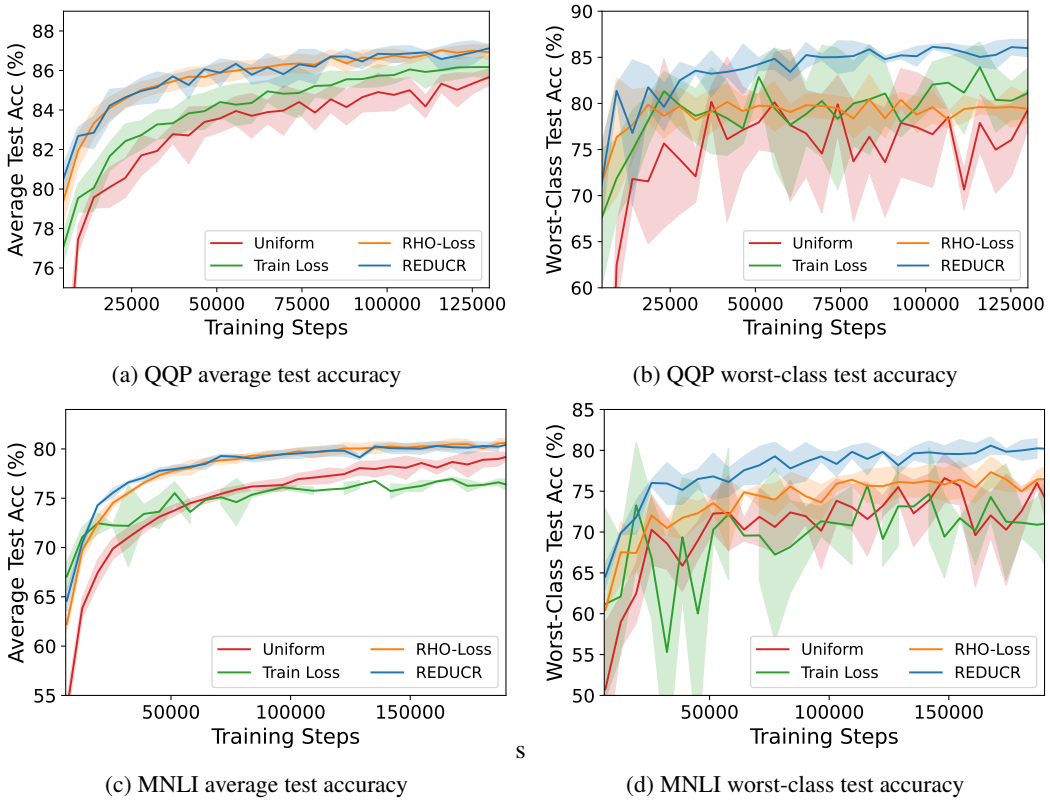
A.7.1 CLOTHING1M TRAINING WEIGHTS

The Clothing1M dataset is imbalanced with respect to class 4. Figure 7 shows that REDUCR is able to consistently identify and weight the underrepresented class across model runs.

A.7.2 CLIPPED EXCESS LOSS ABLATION EXPERIMENTS

To further understand the effects of clipping in the algorithm we analyse the selection score of the selected points with and without clipping. As detailed in Appendix A.4 the class-holdout loss only affects the selection of points via the weights $\mathbf{w}_t$ at each time step, as such we record only the excess loss (the difference between the model loss and class irreducible loss). Figure 8 shows the quantiles of the weighted sum of the excess losses of points selected at each training step for the non-clipped and clipped model respectively. When the excess loss is clipped, Figure 8a shows the selection scores smoothly decrease throughout training as the model loss improves. Without clipping the excess loss decreases smoothly at the beginning of training and then shows unstable behaviour across runs later in training.

In practice we select multiple points per batch by selecting the points with the top k selection scores. When multiple points have the same score, points are selected at random. We note that the clipping does not reduce the excess loss of the selected points to zero where points would be selected randomly to make up the batch.



(a) QQP average test accuracy

(b) QQP worst-class test accuracy

(c) MNLI average test accuracy

(d) MNLI worst-class test accuracy

Figure 6: REDUCR improves the worst-class test accuracy on the MNLI and QQP text datasets whilst maintaining strong average test accuracy performance when compared with the TRAIN LOSS, RHO-LOSS and UNIFORM baselines. On both datasets REDUCR matches the next best performing baseline's mean result across runs approximately 100k training steps earlier.

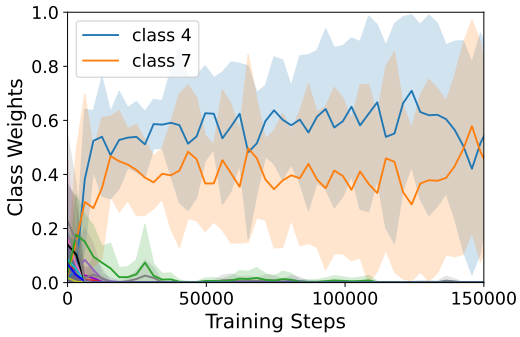
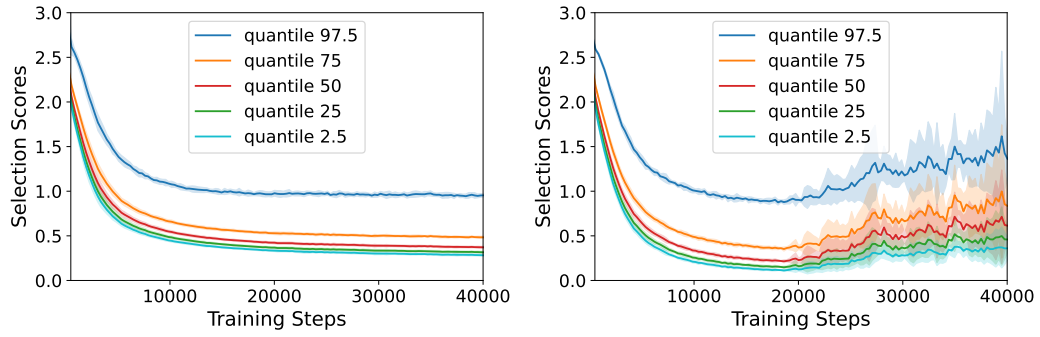


Figure 7: Clothing1M class weights

Intuitively we posit that the clipping reduces the effect of clashing amortised class irreducible loss models in the weighted sum across the $|C|$ selection rules. The amortised class irreducible loss models are trained such that they are an expert in a specific class c. In some cases a model being an expert in a specific class c' may result in it being a poor predictor of classes $C \setminus c'$. Even if this expert has a small weight $w_{t,c'}$ large losses may still propagate into the selection of points. Clipping the excess loss prevents a point from being down-weighted in the weighted sum of class specific selection scores by a specific class too much.



(a) REDUCR Excess Loss

(b) REDUCR Excess Loss No Clipping

Figure 8: The quantiles of the excess loss of points selected at each training step with (a) and without clipping (b) of the excess loss term

A.8 Amortised Class Irreducible Loss Models

In Figure 9 we compare the average expert class ($c \in C$) test accuracy and non-expert class ($c' \in C \setminus \{c\}$) test accuracy across different values of γ for the amortised class-irreducible loss model train on CIFAR10. For the model to be an expert in one class it loses performance in the non-relevant classes. To avoid the problems described in Appendix A.7.2 we selected $\gamma = 9$ for the image datasets as the performance of the non-expert class did not suffer too much.

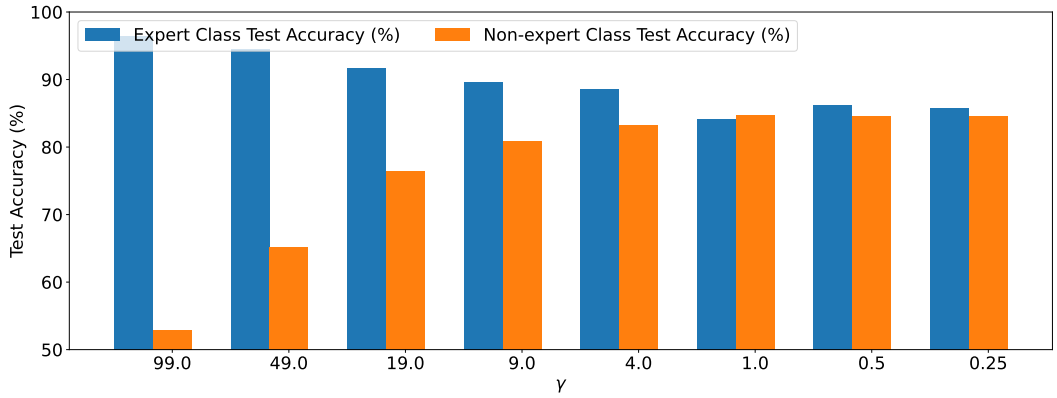


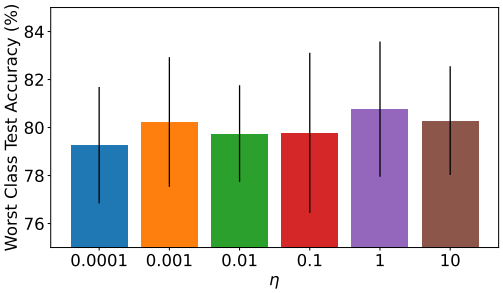
Figure 9: Class-irreducible loss model test accuracies on the expert class and non-expert classes. Class-irreducible loss models are trained using gradient weights $\gamma \in \{0.25, 0.5, 1.0, 4.0, 9.0, 19.0, 49.0, 99.0\}$.

A.9 Hyper-parameter Tuning

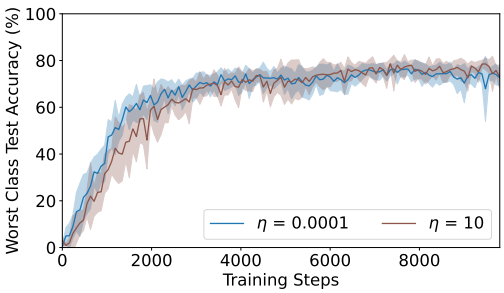
In this section, we perform sensitivity analyses for hyper-parameters introduced by REDUCR. In particular, we investigate the sensitivity of target model performance to the learning rate η used for target model training, the gradient weight γ used for class-irreducible loss model training, the fraction of datapoints selected for target model training n_b/n_B (for a constant selected batch size n_b), and the frequency with which class hold-out losses are updated during target model training.

All experiments in this section use the CIFAR10 dataset. We use ResNet-18 target models (trained using $\eta = 10^{-4}$ and with a percent train of 0.10) and ResNet-18 class irreducible loss models (trained using $\gamma = 9$) unless otherwise stated.

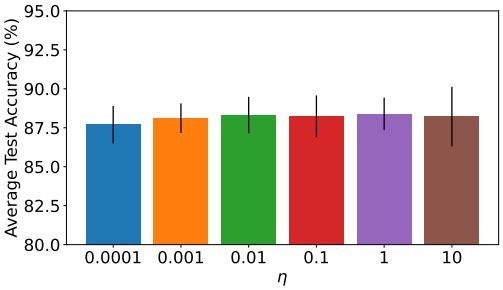
We find that REDUCR's performance is not sensitive to the learning rate η and the frequency of class hold-out loss updating. We find that REDUCR's performance is sensitive to the gradient weight γ, though this is because a larger gradient weight increases the variance of loss gradients and slows model training. REDUCR's performance is not sensitive to the gradient weight at smaller



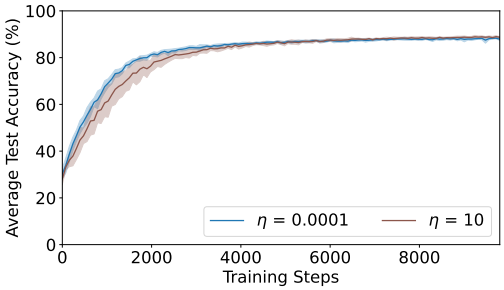
(a) CIFAR10 Final Worst-Class Test Accuracy



(b) CIFAR10 Worst-Class Test Accuracy Curves



(c) CIFAR10 Final Average Test Accuracy



(d) CIFAR10 Average Test Accuracy Curves

Figure 10: **Final average and worst-class test accuracy are not sensitive to the value of η on the CIFAR10 dataset, though both increase faster during training when using smaller values of η.**

gradient weights, for which class irreducible loss model training losses have converged. Finally, we find that REDUCR's performance is not sensitive to the fraction of data points selected for target model training (referred to as the percent train) for intermediate values of percent train, though performance is poor for very low fractions and very high fractions (recall a fraction of 1.0 recovers uniform selection).

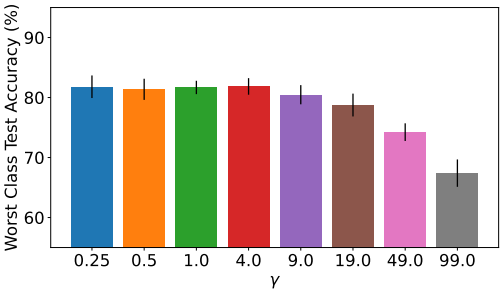
In summary, REDUCR's performance is largely insensitive to the values of newly-introduced hyper-parameters on the CIFAR10 dataset. Sensitivity analyses on additional datasets are needed to increase the robustness of these findings. However, a gradient weight of $\gamma = 9$ and a percent train of 0.10 perform well without additional hyper-parameter tuning for several datasets, class irreducible loss model architectures and target model architectures in our main experiments, which tentatively suggests the robustness of these findings.

A.9.1 Learning Rate η

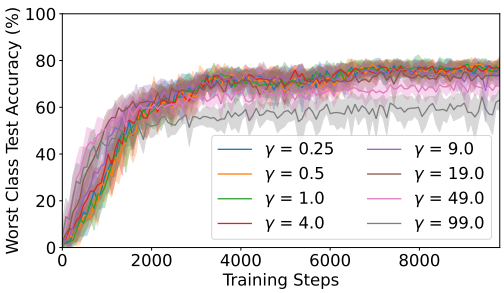
First, we perform a sensitivity analysis on the learning rate η. We train target models using REDUCR for each $\eta \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$.

Experimental results demonstrate that smaller values of η result in a faster improvement in target model performance during training, though final target model performance is similar for all values of η investigated (see figure). Intuitively, larger values of η result in more concentrated class weights. REDUCR therefore uses a more concentrated weighted average of class irreducible losses during datapoint selection, which reduces the quality of selected datapoints. Larger values of η also result in faster changes in class weights between training steps, which reduces the coherence of datapoint selection between training steps.

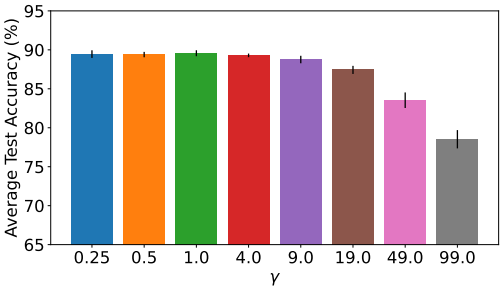
In practice, appropriately small values of η should be used in order to reduce computational cost. Note that what constitutes an appropriately small value of η depends on the scale of losses in a particular domain. Initial target model training runs are therefore necessary to identify a value of η for which class weights do not prematurely concentrate on one class η.



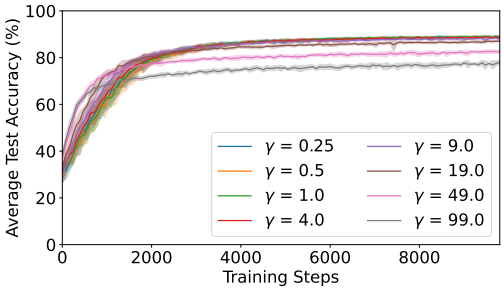
(a) CIFAR10 Final Worst-Class Test Accuracy



(b) CIFAR10 Worst-Class Test Accuracy Curves



(c) CIFAR10 Final Average Test Accuracy



(d) CIFAR10 Average Test Accuracy Curves

Figure 11: **Average and worst-class test accuracy are sensitive to the value of γ on the CIFAR10 dataset**, though this likely reflects longer convergence times for class-irreducible loss model training when using larger values of γ.

A.9.2 Gradient Weight γ

Next, we perform sensitivity analysis on the gradient weight γ. We train sets of class-irreducible loss models for each $\gamma \in \{0.25, 0.5, 1.0, 4.0, 9.0, 19.0, 49.0, 99.0\}$ and train target models trained using REDUCR for each set of class-irreducible loss models.

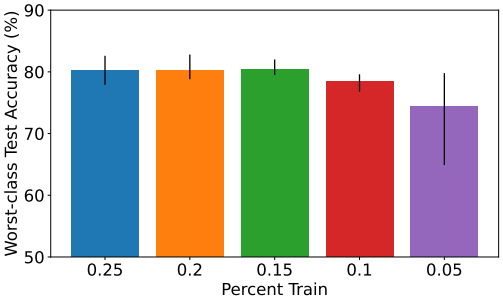
Experimental results show that increases in the gradient weight above 9.0 result in faster improvement in target model performance early in training, though target model performance converges to a lower value later in training for larger gradient weights $\gamma \in \{19.0, 49.0, 99.0\}$.

Higher gradient weights increases the variance of loss gradients, which requires a greater number of training epochs for class-irreducible loss model training. In our experiments, class irreducible loss models trained with gradient weights $\gamma \in \{19.0, 49.0, 99.0\}$ do not converge before the end of training. Poor target model performance for larger gradient weights $\gamma \in \{19.0, 49.0, 99.0\}$ is therefore the result of pre-convergence class-irreducible loss models.

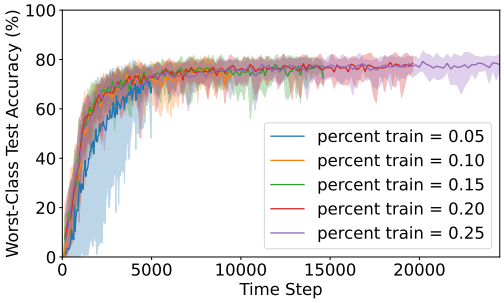
This finding highlights the trade-off between fast target model training (which requires a large gradient weight) and fast class irreducible loss model training (which requires a smaller gradient weight). Regardless, final target model performance is similar providing class irreducible loss models reach convergence, as is the case for gradient weights $\gamma \in \{0.25, 0.5, 1.0, 4.0, 9.0\}$.

A.9.3 Fraction of Selected Datapoints

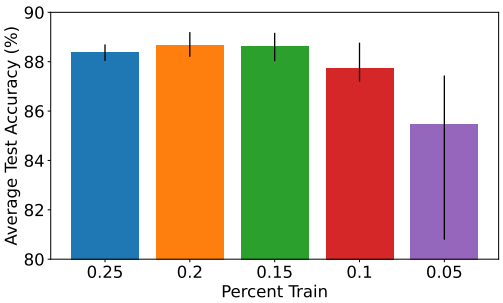
We next perform sensitivity analysis on the fraction of datapoints selected for target model training n_b/n_B (referred to as the percent train). In particular, we use a constant selected batch size n_b and vary the original batch size n_B in order to vary the fraction of datapoints selected for target model training. Intuitively, a smaller percent train allows REDUCR to select from a greater number of candidate datapoints at each training step, which results in the selection of datapoints with larger weighted reducible loss all else the same. Since datapoints with larger (weighted) reducible loss are those from which a model can learn the most (Mindermann et al., 2023), we expect a smaller percent train to result in a faster improvement in target model performance.



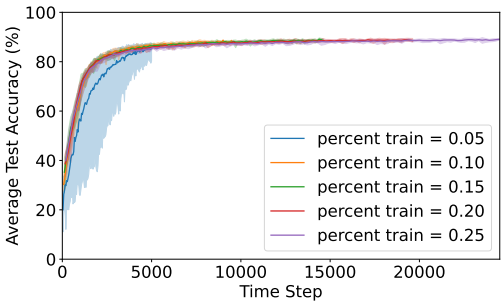
(a) CIFAR10 Final Worst-Class Test Accuracy



(b) CIFAR10 Worst-Class Test Accuracy Curves

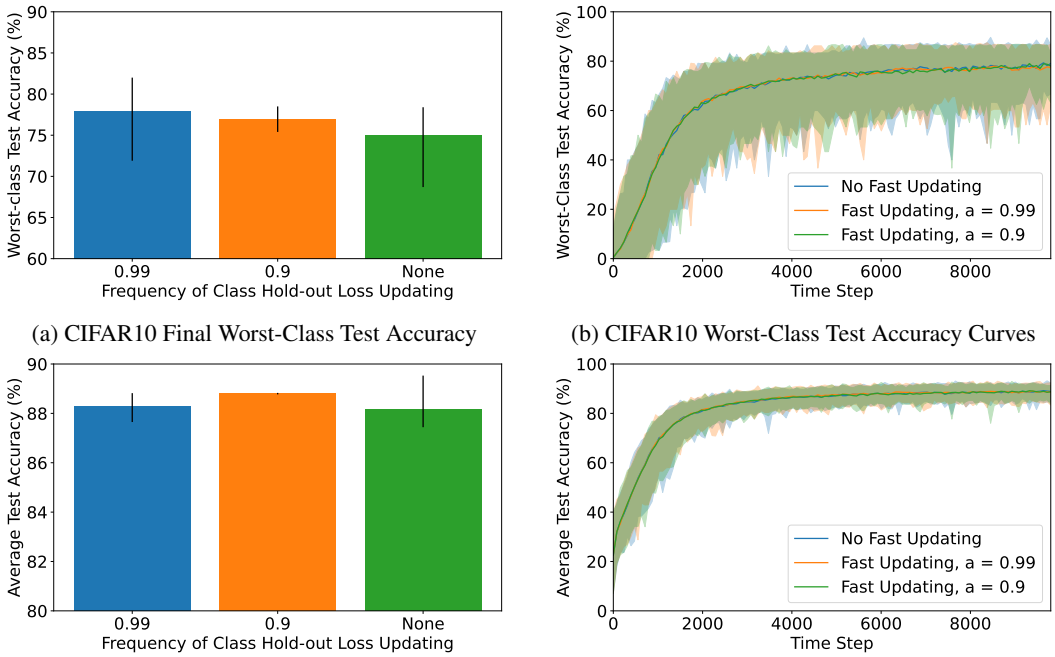


(c) CIFAR10 Final Average Test Accuracy



(d) CIFAR10 Average Test Accuracy Curves

Figure 12: Test accuracies are not sensitive to the value of percent train for intermediate values of percent train. For a percent train of 0.05, the final test accuracy is less than larger percent trains, and test accuracy increases more slowly during target model training. Plots show an average and two standard deviations across 10 seeds. Note that at each training step, targets models have performed the same number of gradient steps, though they have seen different numbers of candidate datapoints. Since each percent train uses a different batch size, training epochs consist of a different number of training steps for each percent train. It is therefore particularly important to compare target models at each training step instead of training epoch. Test accuracies are computed at the end of each training epoch using the current target model and plotted at the training step at the end of the training epoch.



(a) CIFAR10 Final Worst-Class Test Accuracy

(b) CIFAR10 Worst-Class Test Accuracy Curves

(c) CIFAR10 Final Average Test Accuracy

(d) CIFAR10 Average Test Accuracy Curves

Figure 13: Test accuracies are not sensitive to the frequency with which class hold-out losses are updated. Plots show minimums, medians and maximum across 10 seeds.

We train target models using REDUCR for each percent train, batch size pair in $\{(0.05, 640), (0.10, 320), (0.15, 216), (0.20, 160), (0.25, 128)\}$ (i.e., we train all target models using a selected batch size of 32). We find that a percent train of 0.05 attains lower final worst-class and average test accuracy, despite having most candidate datapoints to select from. This is surprising and is in contradiction with the intuition provided above. Furthermore, percent trains $\{0.1, 0.15, 0.2, 0.25\}$ attain similar average test accuracy at the end of training, though larger percent trains attain slightly higher worst-class test accuracy at the end of training.

Experimental results demonstrate the performance of REDUCR is largely insensitive to the percent train (for a constant selected batch size) for non-extreme percent trains (recall uniform selection corresponds to a percent train of 1.0). Therefore, in practice, a selected batch size n_b should first be chosen such that loss gradient estimates have a low variance, and then a batch size n_B should be chosen such that the percent train n_b/n_B is an intermediate value (e.g., 0.10).

Experimental results also suggest that selecting datapoints with the very largest weighted reducible loss for model training may not be most appropriate for improving model performance. Instead of top-k selection, some form of soft selection may result in better model performance.

A.9.4 Frequency of Class Hold-out Loss Updating

In our experiments, class hold-out losses are updated at the end of each training epoch using the full hold-out set. However, target model performance may improve on some classes significantly more than others during a training epoch (especially early in training). We therefore perform a sensitivity analysis on the frequency with which class hold-out losses are updated.

It is computationally expensive to update class hold-out losses using the full hold-out set. To reduce the computational cost of more frequent class hold-out loss updating, it is therefore necessary to update class hold-out losses using only a small subset (e.g., a single batch) of the full hold-out set at each update. However, class hold-out losses computed using a small subset of hold-out set datapoints are noisy. We therefore use an exponentially-weighted moving average of class hold-out losses computed on batches of hold-out set datapoints. In particular, a batch of size n_B is sampled uniformly at random from the hold-out set at each training step. Losses are then computed for each datapoint in the sampled batch using the current target model. Finally, for each class $c \in [C]$,

losses of datapoints of class c in the sampled batch are averaged and used to update a de-biased exponentially-weighted moving average with decay parameter $a \in [0, 1]$.

We perform experiments using exponentially-weighted moving averages with decay parameters $a \in 0.9, 0.99$ for fast updating of class hold-out losses. Experimental results demonstrate REDUCR's performance is not sensitive to the frequency of class hold-out loss updating.

A.10 Highly Imbalanced Datasets

We also conduct experiments with 0.25% and 0.5% percent imbalances on classes 3 and 5. However, (class) irreducible loss models and target models only receive 6.25 and 12.5 datapoints of the imbalanced class during one training epoch (in expectation) with percent imbalances of 0.25% and 0.5% (respectively). As a result, too few datapoints of the imbalanced class are seen during model training to achieve good performance on the imbalanced class.

This demonstrate a failure mode of existing selection methods. During target model training, datapoint selection cannot improve performance on the imbalanced class in the presence of severe under-sampling, since datapoints of the imbalanced class are not sampled sufficiently often (before selection). Additionally, for selection methods that use some form of reference model, severe under-sampling also affects reference model training. An explicit correction for severe under-sampling is needed during reference model training and target model training (e.g., an importance sampling correction or a replay buffer similar to that used in reinforcement learning).