Knowledge and Information Systems (2025) 67:11161–11191
https://doi.org/10.1007/s10115-025-02563-7

RESEARCH

Focal Sampling: SGD biased towards early important samples for efficient image classification with augmentation selection

Alessio Quercia[1,4] · Fernanda Nader[1] · Abigail Morrison[2,4] · Hanno Scharr[1] · Ira Assent[1,3]

Received: 9 July 2024 / Revised: 22 July 2025 / Accepted: 24 July 2025 /
Published online: 28 August 2025

Abstract
In deep learning, using larger training datasets usually leads to more accurate models. However, simply adding more but redundant data may be inefficient, as some training samples may be more informative than others. We propose Focal Sampling, a method that biases SGD (Stochastic Gradient Descent) towards samples that are found to be more important after a few training epochs, by sampling them more often for the rest of the training. In contrast to state-of-the-art, our approach requires less computational overhead to estimate sample importance, as it computes estimates once during training using the prediction probabilities, and does not require restarting training. In the experimental evaluation, we see that our learning technique trains faster than state-of-the-art and can achieve higher test accuracy, especially when datasets are not well balanced or when using multiple data augmentations. Lastly, results suggest that our approach has intrinsic balancing properties and that balancing datasets based on class importance, rather than by number of samples, can achieve higher test accuracy. Code is available at https://jugit.fz-juelich.de/ias-8/sgd_biased.

Keywords Deep Learning · Optimization · Hard Example Mining

This work was funded by the Helmholtz School for Data Science in Life, Earth, and Energy (HDS-LEE).

✉ Alessio Quercia
a.quercia@fz-juelich.de

Fernanda Nader
f.nader.nieto@fz-juelich.de

Abigail Morrison
a.morrison@fz-juelich.de

Hanno Scharr
h.scharr@fz-juelich.de

Ira Assent
i.assent@fz-juelich.de

1 Institute for Advanced Simulation, IAS-8: Data Analytics and Machine Learning, Forschungszentrum Juelich, Juelich, Germany

2 Institute for Advanced Simulation, IAS-6: Computational and Systems Neuroscience, Forschungszentrum Juelich, Juelich, Germany

3 Department of Computer Science, Aarhus University, Aarhus, Denmark

4 Department of Computer Science, RWTH Aachen University, Aachen, Germany

1 Introduction

For many deep learning problems, increasing the model and training data sizes boosts the model performance and ability to generalize [1, 2]. However, this increase comes at the ever higher cost of long training times and great energy consumption. One way to reduce this cost is to optimize the model training procedure. The most common training approach relies on random shuffling without replacement of the data samples during training for a given amount of epochs. As a consequence, all the samples are seen by the model the same number of times and are therefore implicitly treated as equally important.

There is increasing interest in understanding the importance of training samples for model generalization to unseen data, in order to boost its performance. Existing methods generally face two main challenges. The first group of methods imposes some computational overhead for pre-computation of sample importance. The second group introduces overhead to update the importance scores during training and hamper parallel data loading when computing importance in an online fashion.[1] This reduces their benefit, in particular for datasets requiring expensive reading operations and/or transforms.

Methods in the first group prioritize important samples at the expense of additional overhead for determining sample importance. *Data pruning* [3–9] ranks all training samples in a dataset according to an importance metric. They show that less important samples can be pruned before training with little to no loss in test accuracy. However, this approach usually entails fully training at least one model, computing and ranking the sample scores, and then selecting a subset of samples that well approximates the test accuracy on the full training dataset. Similarly, other methods require training a model [10, 11] or solving a linear programming problem [12] before training starts.

Methods in the second group such as *importance sampling*, *hard example mining*, *curriculum learning*, and *coresets selection* speed up training by online sampling proportional to gradient norms via an unbiased estimator [13–15] (or approximations [16–21]), biasing training to hard samples [22–30], presenting samples through a curriculum [31–34], or adaptively subsetting training data to approximate the full gradient [35–38].

As detailed in Sect. 6, additional computational overhead to compute sample importance may render methods relying on them less beneficial in practice, especially for large datasets. This is particularly true for methods that require the computation of per-sample gradients (currently prohibitively expensive [20]), training additional networks, or running expensive operations prior or during training.

We address these challenges by proposing *Focal Sampling*, a simple, yet effective SGD-based method that estimates sample importance *once early in training*, as in [8], and samples proportionally to it for the remaining iterations, biasing the model towards samples that are more important for *this model*. By doing so, we avoid the computational disadvantages of both groups of methods, as we do not need to restart training or update sample importance estimates online. Please note that, unlike *importance sampling* methods, we use a *biased* estimator: we expect datasets to be imbalanced in terms of importance, thus re-balancing in this regard improves performance.

As the empirical evaluation in Sect. 3 shows, our approach outperforms data pruning [8], importance sampling [17], hard example mining by re-weighting the loss [25], prioritizing samples with high prediction variance [26], subsampling from larger batches at each iteration [27], achieving high accuracy at greatly reduced cost.

[1] Methods updating the data loader's sampling distribution after every iteration cannot leverage batch prefetching, as sampling the next batch requires the current one to finish first.

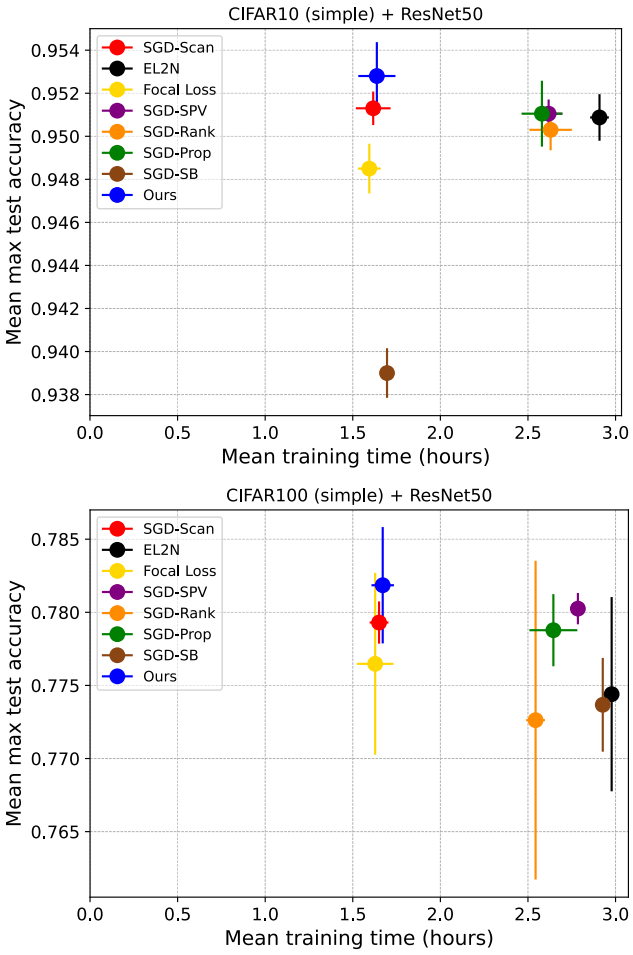

Fig. 1 Mean maximum test accuracy with respective standard deviation vs. mean compute time for different training algorithms of ResNet50 on CIFAR10 (top) and CIFAR100 (bottom). Upper left is better. Ours is the only method that outperforms SGD at comparable runtime

As shown in Fig. 1 (details in Sect. 3), our method is the only one to deliver higher average maximum accuracy with respect to standard training (SGD-Scan) at little to no additional overhead, where state-of-the-art needs considerably more training time to achieve similar accuracy. We observe, here and consistently in all our experiments, that our method yields competitive or improved results even when Focal Loss reduces performance, despite the fact that the underlying weight function is identical. This may be due to the fact that weighting the loss function tampers with the effective learning rate, influencing convergence, while our sampling approach keeps the loss untouched.

Our main contributions [39] include:

- showing that a model can determine a sample importance distribution early in training and use it as sampling distribution for the rest of training to improve learning;

- identifying sample importance at reduced computational cost with respect to state-of-the-art methods, as it is computed once during training and it does not require training to be restarted;
- a method that much faster reaches the maximum mean accuracy achieved by uniform sampling;
- a method that reaches higher accuracy than baselines, when given the same time budget;
- showing that our method converges faster and with higher accuracy on CIFAR10, CIFAR100, iNaturalist2021, and with comparable performance on ImageNet-1K;
- automatic balancing of class distributions and empirical observations in favour of balancing class distributions by class or sample importance, rather than by number of samples;
- a method that automatically selects most informative samples from multiple augmentation sets, obtaining significant speedup and better performance;
- empirical evidence that sample importance shows some model, initialization and task dependency, raising the important question in contrast with data pruning methods: "can samples in a dataset be ranked by importance in a model-independent manner?";
- hypotheses and observations on sample importance, empirically showing that the sooner the importance weights are computed and applied, the greater the benefits;

2 Our method

Standard training of deep learning models makes the implicit assumption that each training sample is equally important in a uniform scan through all samples in every epoch. Each sample is used exactly the same number of times (i.e. the number of epochs). In contrast, we hypothesize that a sample can be more or less important for a given model to learn a given task, and that, in contrast to previous work, we can estimate sample importance *efficiently during training* to speed up learning by focusing on more important samples.

2.1 Learning effectively and efficiently

We identify sample importance as sample difficulty: the harder a sample, the more a model can learn from it to improve generalization. Conversely, a sample that can be easily classified can be considered already learned and is, therefore, less informative.

Let $S = (x_i, y_i)_{i=1}^{M}$ be a training dataset with M samples, $V = (x_i, y_i)_{i=1}^{L}$ a test dataset with L samples, where $x_\cdot$ are sample feature values and $y_\cdot$ one-hot encoded class labels. Further, let $f_\theta(.)$ be a model (e.g. a neural network) that can be trained using a gradient descent optimizer (e.g. stochastic gradient descent, SGD), θ its parameters, $f_\theta(x_i)$ the output probabilities after *softmax* activation. Additionally, let N be the number of training epochs and B the training batch size, and let

$$P(i|S_e, S) = 1/(|S| - |S_e|)\mathbf{1}_{i \notin S_e} \tag{1}$$

be the standard training sampler that uniformly scans through all the samples at every epoch without replacement [26], where $\mathbf{1}$ is an indicator function and S_e is the set of samples already selected during the current epoch. Let $\mathcal{L}$ be a training loss function (here $\mathcal{L}(f_\theta(x_i), y_i) = -y_i \log(f_\theta(x_i)_{y_i})$, i.e. the Cross-Entropy loss). The gradient step at iteration

t can be computed as

$$\theta_t = \theta_{t-1} - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta_{t-1}} \mathcal{L}(f_{\theta_{t-1}}(x_i), y_i) \tag{2}$$

with learning rate η and (x_i, y_i) sampled from S using the sampler $P(i|S_e, S)$.

Let $A(V, f_{\theta_t}(.))$ be the accuracy of model f_{θ_t} at training time t on all test samples $(x_i, y_i) \in V$. Then, let $A_b = \max_t(A(V, f_{\theta_t}(.)))$ be the highest test accuracy achieved by the model $f_\theta(.)$, and $t_b = \arg\max_t(A(V, f_{\theta_t}(.)))$ be the training time needed to reach A_b using standard sampling (1). The 'speedup factor' of a training method yielding model $f_{\theta'}(.)$ (with $'$ indicating a different training method) is given as $1 - \min(t_n/t_b, 1)$, if it reaches baseline accuracy A_b at time t_n:

$$t_n = \min_{t \leq t_b}(t) \quad \text{s.t.} \quad A(V, f_{\theta'_t}(.)) \geq A_b \tag{3}$$

Informally, given a training and a test set, a model, an optimizer, and a baseline maximum accuracy, we search for the minimum training time to reach this accuracy. Note that we choose accuracy as metric to assess the generalization of the trained model, but Eq. (3) can be extended to other metrics of model performance.

2.2 Biasing SGD towards important samples

We hypothesize that a model can learn a given task faster or more reliably by focusing on important samples in a given dataset. To find sample importance, we leverage the concept of focus during training used for Focal Loss [25], but from a different perspective: instead of regularizing the loss based on sample difficulty $(1 - f_\theta(x_i)_{y_i})^\gamma$, we define a sampler P assigning each training sample (x_i, y_i) a probability of being selected according to its difficulty:

$$P(i|f_{\theta_E}, S) = \frac{w_i}{\sum_{j=1}^{M} w_j} \tag{4}$$
$$\text{with} \quad w_i = (1 - f_{\theta_E}(x_i)_{y_i})^\gamma \quad \text{and} \quad (x_i, y_i) \in S$$

P uses replacement and ensures that already learned samples (easy samples) are downsampled, whereas samples that are still to be learned (hard samples) are oversampled. $f_{\theta_E}(x_i)_{y_i}$ is the probability that model f_{θ_E} classifies the training sample x_i in the correct class y_i, f_{θ_E} is the model pretrained using the uniform sampler without replacement (1) for E epochs. Lastly, γ is the *focusing* parameter that controls the focus to put on hard examples. So, $\gamma > 0$ emphasizes hard samples, $\gamma < 0$ emphasizes easy samples and $\gamma = 0$ treats all training samples as equally important.

We propose a method that biases SGD towards important samples computed early in training:

1. Pretrain a model f on all training samples for $E \ll N$ epochs using the standard sampler (1).
2. At the end of epoch E, use the model predictions to update the sampler as described in (4).
3. Train f for $N - E$ epochs using the new sampler, i.e. focusing on more important samples.

2.3 Automatic augmentation set selection

It is well known that using augmentations during training increases the difficulty and variety of the samples, making the model more robust during inference. However, it is a major challenge to identify the most beneficial augmentations for training. In this regard, we propose a method that can quickly identify the most useful augmentations to learn a given task, saving time on less informative augmentations, and therefore converging faster than baselines. In particular, we repeat a given dataset for multiple different augmentation sets, resulting in a larger and more varied dataset. Our method is able to identify the augmentation sets producing the most beneficial samples, and samples more often from it, converging faster than other baselines, which spend more time in learning less beneficial augmented samples.

Formally, we extend our method to a scenario where D augmentation sets are used for all training samples M, resulting in a dataset $S_D = ((x_{ij}, y_{ij})_{i=1}^{M})_{j=1}^{D}$ with $M \times D$ samples. By doing so, not only we prioritize important samples, but also harder augmentation sets on these samples. This increases the overall variety and difficulty of every sample. On one hand, using multiple augmentation sets gives easy samples a chance to become harder and gain priority, on the other hand it gives hard samples a chance to get sampled even more with multiple variations, therefore making them easier to learn.

2.4 Method analysis

a) Why early epoch E?: It has been empirically shown that training dynamics stabilize after an early chaotic stage and it is harder to improve the learning process in later stages [40]. Therefore, estimating the importance weights too early in training would lead to an inaccurate sample importance distribution. On one hand, we hypothesize that the accuracy of the sample importance distribution is directly proportional to the training time, i.e. the longer a model is trained, the better its sample importance estimates. On the other hand, we hypothesize that the benefits of using an importance distribution during training are inversely proportional to the training time, i.e. the sooner the importance weights are applied, the greater their benefit. We further discuss these hypotheses in Sect. 4.4. Other works [8, 33, 41–43] show that it is possible to distinguish between easy and hard samples early in training. Inspired by these concepts and by the data pruning work of Paul et al. [8], which shows that it is possible to prune a dataset early in training without losing performance, we compute sample importance once after a few training epochs E. This parameter represents the optimal trade-off between training long enough to have an accurate importance distribution and applying the weights early enough to have benefits during training. Unlike [8], we compute the importance weights using a single model instance (instead of ten), and do not train from scratch once the sample importance is obtained. By doing so, we ensure that no additional training iterations are required to estimate sample importance. Moreover, this allows our method to save time with respect to online methods, which update the importance during training. In practice, this also allows our method to be applied to real-world offline datasets as it does not suffer from fetching overhead.

b) Importance metric choice: We use the importance term w_i used by Focal Loss [25] to compute sample importance weights (4), but from a different perspective. Focal Loss computes w_i at every iteration and uses it as a regularization weight for the loss, while considering samples equally important from the point of view of the sampler. Our method, in contrast, computes the importance distribution once early in training and then samples hard samples more often than easy samples, without adding a regularization weight to the

loss. Note that adding a weight is still possible, and therefore loss weighting methods can be seen as complementary methods. Additionally, please note that we choose (4) as importance metric as it performed best in comparison with other metrics in preliminary experiments, e.g. loss, EL2N [8] and gradient norms metrics (see Sect. 5.1). Choosing the optimal importance metric is complementary, and out of scope for the current study.

c) Why multiple augmentations?: Choosing the right augmentations for a given dataset and task is a major challenge in computer vision. To circumvent this problem, we choose multiple augmentation sets and let our method learn an implicit importance distribution over the augmentation sets, automatically giving more weight to the ones producing the most beneficial augmented samples. By quickly doing so, our method achieves a significant speedup, and a boost in performance.

d) Hyperparameter tuning: We tune E and γ on CIFAR10 and CIFAR100 (see Appendix A). We find that our method is robust with respect to E. In all tested cases, 5 to 20 pretraining epochs are enough to estimate sample importance and achieve better or comparable results than SGD. In some cases even one or two are sufficient. Additionally, we notice from experiments that a choice of $E = 10$, tuned on CIFAR10, is also a good choice on the much bigger iNaturalist2021, suggesting that this parameter setting may be a good initial choice across datasets. Thus, we use $E = 10$ in all our experiments, unless stated differently. With E fixed, we tune γ and find that values in the range 0.1-2 generally improve results. We select $\gamma = 0.5$ for all our experiments.

e) Why bias training?: We assume that a given dataset can have a difficulty imbalance, i.e. samples/classes that are more difficult than others. With this assumption in mind, we seek to balance the model training by prioritizing harder samples. In practice, differently from *importance sampling* methods, we want to bias training towards hard samples, therefore we do not scale the gradient steps by the inverse of the sample weight used for sampling it. Additionally, please note that [8] biases training using a step function as sampling distribution (i.e. 0 for pruned samples and uniform in the new data length for the remaining ones) when training from scratch, whereas our method generates a probability distribution out of the sample importance scores.

f) Model, initialization and task dependency: We investigate the following important question "can samples in a dataset be ranked by importance in a model-independent manner?". In contrast with data pruning methods, we hypothesize that importance is model, initialization and task dependent, and therefore that a sample importance ranking should be computed for each model architecture, initialization and task for any dataset. From our experiments it emerges that sample importance is model and task dependent, whereas only mildly initialization dependent. This potentially suggests the sample scores (or importance) computed once by data pruning approaches would need to be re-computed not only for each new dataset, but also for each new model architecture, model initialization and task, limiting their re-usability. Further discussion in Sect. 4.

3 Experimental results

Our baseline is standard SGD with uniform sampling without replacement (see (1), *SGD-Scan*). We compare against the main representatives of the methods outlined in the Introduction (Sect.1), and discussed in more detail in Related Work (Sect.6). The closest related methods are *Focal Loss* [25] and data pruning with *EL2N* scores [8]. For the former, we use the implementation available at [44] and set $\gamma = 2$, as in the original work, and as it

yields the best results among different tested values (0.5 and 3 [45]). For the latter, we use the pre-computed scores provided to us by the authors of [8] (computed using ten models trained for 20 epochs) and pruned the lowest 30% of CIFAR10 and 10% of CIFAR100. Please note that this method requires 78000 additional iterations (7800 per model instance) before training starts to compute the sample importance scores.

Additionally, we compare against the sampling method based on sample prediction variance proposed by Chang et al. [26] (*SGD-SPV*), which emphasizes samples with high prediction variances, and the two loss-based prioritization methods proposed in [17], *proportional* and *rank-based* (respectively, denoted *SGD-Prop* and *SGD-Rank*). In particular, for *SGD-SPV* we use ten *burn-in* epochs for CIFAR10 and 50 for CIFAR100, and $\epsilon = 0.2$, as described in the paper.

Lastly, we compare against *Selective-Backprop* (*SGD-SB*, [27]), which computes multiple forward passes before running a backward pass on samples with high losses. As suggested in the original work, we use queue size $r = 128$ (as the batch size) and *selectivity* of 33% ($\beta = 2$) on CIFAR10 and 50% ($\beta = 1$) on CIFAR100. Note that *SGD-SB* is similar to [11, 20, 29], however [27] showed superior performance to [20], whereas [11, 29] require additional computational overhead, during and before training, respectively.

Except for *Focal Loss* and *EL2N* scores pre-computation, all methods are implemented in our own pipeline to ensure high efficiency and fair comparison.

We run all experiments four times (with predefined random seeds) and report mean results and standard deviations (or 16th and 84th percentiles). As detailed in Sect. 6, we do not consider methods with prohibitive runtimes.

3.1 Comparative evaluation

We compare our method to the approaches described above using CIFAR10 and CIFAR100 [46]. We use the datasets in the same way as they are used in the work by [8]: 50000 samples for training and 10000 for test. As data augmentation we pad all sides by 4 pixels, random crop to 32x32 pixels and horizontally flip the image with probability 0.5. We here use a ResNet50 with 1x1 stride for the first convolution and without the first max pooling layer. We use the same training hyperparameters used by [8]: Stochastic Gradient Descent (SGD) optimizer, initial learning rate 0.1, Nesterov momentum 0.9, weight decay 0.0005. We use batch size 128 and train for 78000 iterations (i.e. 200 epochs) for both CIFAR10 and CIFAR100. We decay the learning rate after 23400, 46800, and 62400 iterations. In our method, we set the focusing parameter $\gamma = 0.5$ for all experiments as preliminary experiments indicate it is a stable and well-performing parameter setting, as discussed in Sect. 2.4. For CIFAR100 we here use $E = 1$ as it resulted in even better performance than $E = 10$, which was already performing slightly better than SGD-Scan.

Figure 1 shows the mean maximum accuracy achieved by each tested method during training time. We observe that our method is the only one to achieve the goal to outperform the uniform baseline SGD-Scan in both CIFAR10 and CIFAR100 within the same training time , whereas all the other methods fail even when granted some additional training time (i.e. 3 h, approximately double the time required by SGD-Scan). This result is even more pronounced, when requiring to strictly stay within the time needed for SGD-Scan to finish training (see Table 1). Speedups achieved for other datasets are given in Fig. 2 left.

In Fig. 2 (right), we see the training times for each method to train a ResNet50 on CIFAR10 for 200 epochs. Our method runs approximately with the same time as SGD-Scan and Focal

Table 1 Maximum mean test accuracy achieved strictly within SGD-Scan's time budget when using simple and multiple augmentations

Data	Aug	SGD-Scan	Focal Loss	SGD-SPV	SGD-Rank	SGD-Prop	SGD-SB	Ours
CIFAR10	Simple	95.1 ± 0.1	94.8 ± 0.1	94.5 ± 0.3	94.4 ± 0.1	94.4 ± 0.1	93.8 ± 0.1	$\mathbf{95.2 \pm 0.2}$
CIFAR100	Simple	77.8 ± 0.2	77.5 ± 0.6	76.8 ± 0.2	75.9 ± 1.0	76.4 ± 0.2	75.2 ± 0.5	$\mathbf{78.1 \pm 0.4}$
CIFAR10	Multi	95.7 ± 0.2	95.4 ± 0.2	94.8 ± 0.1	94.5 ± 0.1	94.5 ± 0.1	94.1 ± 0.1	$\mathbf{96.2 \pm 0.2}$
CIFAR100	Multi	79.9 ± 0.5	79.1 ± 0.5	78.1 ± 0.6	76.7 ± 0.3	76.1 ± 0.5	75.1 ± 0.3	$\mathbf{80.1 \pm 0.7}$

Note that EL2N is omitted as pruning the dataset requires the same amount of time as for SGD-Scan to complete training

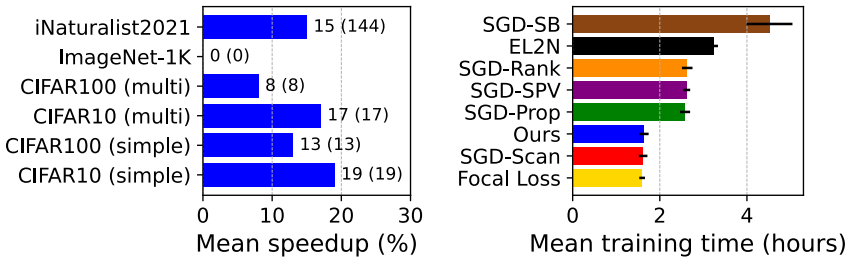

Fig. 2 (Left) Mean speedup (%) and saved time (in parentheses, in minutes) of our method to achieve the maximum mean accuracy of SGD-Scan on different datasets/configurations. (Right) Mean training time required by our method and baselines for 200 epochs on CIFAR10

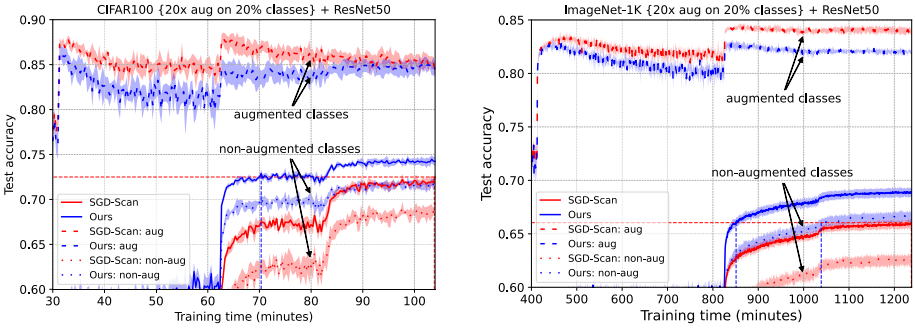

Fig. 3 Test accuracy of all (solid lines), augmented (aug, dashed lines) and non-augmented classes (non-aug, dotted lines) of ResNet50 on CIFAR100 (left) and ImageNet-1K (right) for standard training (*SGD-Scan*) and Ours, where 20% of classes are randomly selected and augmented 20 times. Shaded regions indicate 16th and 84th percentiles of 4 independent runs (i.e. different random seeds). The horizontal red dashed line is average maximum accuracy by standard training (*SGD-Scan*). The vertical dashed lines indicate the first iteration in which our method on average reaches that maximum accuracy

Loss, whereas SGD-SB and EL2N require considerably more time as they, respectively, do multiple forward passes per backward pass and require to prune the dataset first.

3.2 Sample importance has intrinsic balancing properties

As mentioned at the end of Sect. 3.1, our method works best when there is a difficulty imbalance among the samples of a dataset. We validate our hypothesis using a vanilla ResNet50 on iNaturalist2021 [47], an imbalanced dataset with 2686843 images from 10000 object classes for training, 100000 images for validation and 500000 for test. As augmentations we use RandomResizeCrop with size 224, RandomHorizontalFlip and ColorJitter. We train our models using the PyTorch procedure for ImageNet-1K described in [48] with the following changes: the learning rate scheduler is decayed after 30, 60 and 75 epochs, the batch size is 2048 (128 per GPU, with 16 GPUs), and we use mixed precision.

Figure 4 shows that SGD-Scan achieves a test accuracy of 80.4 ± 0.1 in approximately 16h30, whereas our method reaches the same accuracy 2h24 earlier (see Fig. 2, left) and 80.6 ± 0.1 with the same total training time, without requiring tuning of E or γ (same as for CIFAR10), confirming the effectiveness of our method.

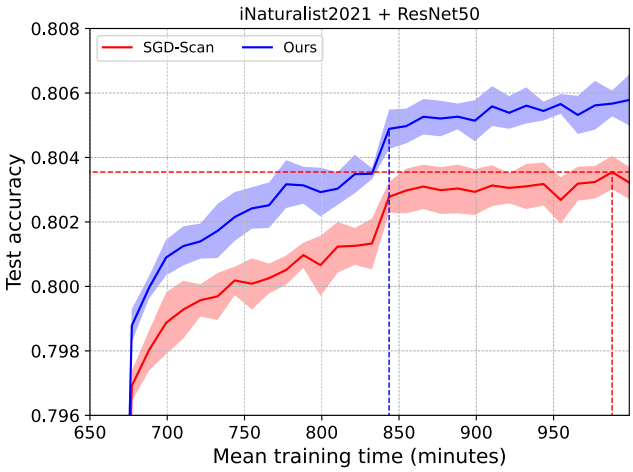

Fig. 4 Test accuracy of ResNet50 on iNaturalist2021 for SGD-Scan and Ours. Solid lines indicate the mean, shaded regions 16th and 84th percentiles of 4 independent runs (i.e. different random seeds). The horizontal red dashed line is average maximum accuracy by standard training (*SGD-Scan*), the vertical dashed lines the first iteration in which our method on average reaches that maximum accuracy

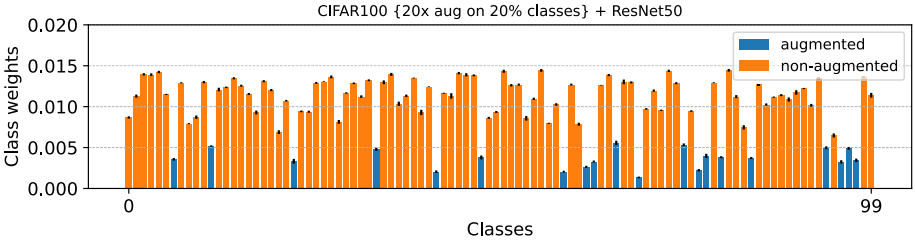

Fig. 5 Class weights implicitly learned by ResNet50 on CIFAR100 where 20% of the classes are randomly selected once and augmented 20 times

To further validate our hypothesis, we manually create imbalance on the difficult CIFAR100, randomly selecting 20% of the classes (once for all different runs) and augmenting them 20 times. In this setting, Fig. 3 (left) shows that our method converges faster and to a better solution with respect to the standard baseline. In particular, our method sacrifices a little accuracy on the augmented classes to focus more on the non-augmented ones, which contain less samples, therefore improving the accuracy on them. This is confirmed in Fig. 5, which shows that our method correctly identifies the augmented classes and assigns them lower class importance weights, therefore prioritizing the underrepresented classes.

These results suggest that our method automatically balances the class distributions by sampling more often samples that appear less, considering them more important for training. Note that we observe very similar results on ImageNet-1K [49] (1281167 training, 50000 validation and 100000 test images from 1000 object classes), where our method (76.3 ± 0.2) achieves comparable results to SGD-Scan (76.3 ± 0.1) on the standard dataset, but yields better results when the class imbalance is manually generated (Fig. 3, right, $E = 2$). Note that we use the same data augmentations and training configuration used for iNaturalist2021, but with the original batch size (256) and without mixed precision. For the experiment in Fig. 3, we use the following set of 5 different augmentations repeated 4 times, where all sets

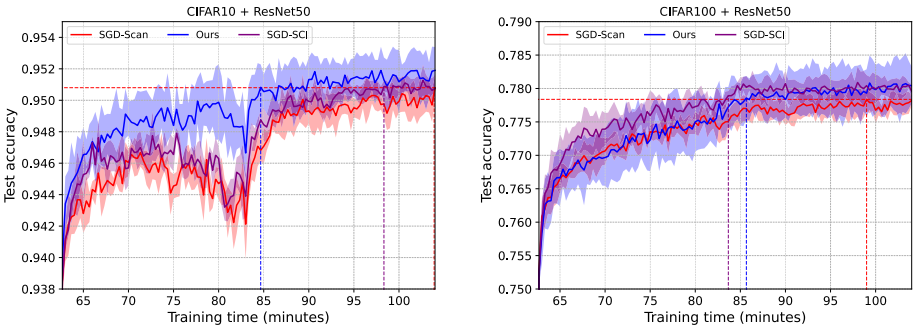

Fig. 6 Test accuracy of ResNet50 on CIFAR10 (top) and CIFAR100 (bottom) using the baseline approach where classes are balanced (SGD-Scan), our approach with variable E and $\gamma = 0.5$, and our class-importance sampled method (SGD-SCI). Solid lines indicate the mean, shaded regions 16th and 84th percentiles of 4 independent runs (i.e. different random seeds). The horizontal red dashed line is average maximum accuracy by standard training (*SGD-Scan*). The vertical dashed lines indicate the first iteration in which a method on average reaches that maximum accuracy

are preceded by RandomResizeCrop with size 224: RandomHorizontalFlip, RandomHorizontalFlip and ColorJitter, AutoAugment [50], RandAugment[2] with parameters suggested in [48] for procedure A3, and TrivialAugmentWide [51].

We conclude that our method works best when there is a difficulty imbalance in the given dataset and that, when samples are (almost) equally difficult, it performs on par with vanilla SGD.

A balanced class distribution is not necessarily the best option Methods like SMOTE [52] try to balance the class distribution in tasks where class imbalance negatively affects model performance. We hypothesize that a class distribution balanced by same number of samples is not necessarily the best option and that classes may better be weighted according to their importance for learning. In Fig. 6, we investigate this by comparing SGD-Scan (evenly balanced on CIFAR10 and CIFAR100), our method and classes sampled according to the *class importance* weights $\bar{w}^c$ (*SGD-SCI*), aggregating sample importance by grouping sample weights w_i (see (4)) by ground truth class $c = y_i$, i.e. $\bar{w}^c = \sum_i w_i|_{y_i=c} / \sum_{j=1}^{M} w_j$.

Note that results shown for SGD-Scan and our method belong to the same runs as Fig. 1.

For both datasets, SGD-SCI on average performs better than the baseline SGD-Scan, suggesting that it is possible to improve model performance on a given task by sampling based on *class importance*, rather than balancing by number of samples per class. However, our method achieves an overall better performance, therefore, we conclude that sampling by *class importance* is a valid alternative to balancing by number of samples per class, but that finer-grained measures (e.g. our sample importance) should be preferred. Intuitively this makes sense, when considering that a class may consist of multiple complicated subclasses, that need some extra care in comparison with 'simple' classes.

3.3 Sample importance with multiple data augmentations

We test our automatic augmentation set selection method extension by designing an experiment where the original dataset is augmented 5 times, resulting in a bigger training dataset

[2] from https://github.com/rwightman/pytorch-image-models.

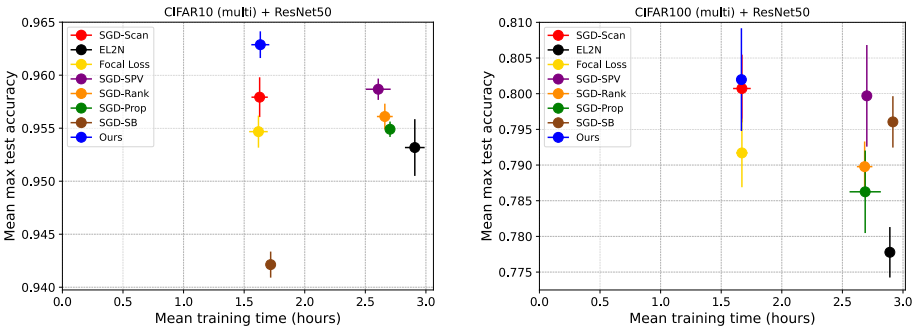

Fig. 7 Mean maximum test accuracy and mean compute time with respective standard deviations of ResNet50 on CIFAR10 (left) and CIFAR100 (right) when using multiple augmentations. Upper left is better

(250000 training samples for both CIFAR10 and CIFAR100). We expect our method to identify the most useful augmentation sets for a given task, and use them to generate harder samples to learn the task faster. Consequently we scale E from 10 to 2 and use it for both datasets. Note that we still load the original images, but they are selected 5 times with 5 different augmentations during training, i.e. twice the simple augmentations described above, AutoAugment [50] with CIFAR10 policy, RandAugment [53] and TrivialAugmentWide [51] with default parameters.

Comparing results for CIFAR10 (simple) and CIFAR100 (simple) with those using multiple augmentations for CIFAR10 (multi) and CIFAR100 (multi) in Table 1 shows that mostly all methods benefit from the augmentations. From Fig. 7 and Table 1, we can see that our method is again the only one to outperform SGD-Scan without additional training time. The speedup with multiple augmentations is only slightly worse compared to simple augmentations (Fig. 2), suggesting that multiple data augmentation yield better results with approximately the same computational effort. Therefore using multiple augmentations can be beneficial as it comes at no significant additional cost, but achieves a higher accuracy.

4 Sample importance analysis

In the following sections, we discuss sample importance's model, initialization and task dependency. Lastly, we report additional hypotheses and observations on the nature of sample importance and how to use this to get a performance boost.

4.1 Model dependency

In Fig. 8, we report class importance weights aggregated from sample importance weights computed by our method with different models and epoch $E = 10$ (top) and the best epoch E for each model architecture (bottom). Comparing the two figures shows that class weights from different models get closer when computed at the same epoch (the mean rank correlation of sample weights from all models is 0.81), but there are still differences between class weights, implying that sample importance evolves in a model-specific way during training.

Figure 10 reports several ablation studies to better show the model dependency of sample importance. In particular, we used the best weights for different model architectures as weights for training ResNet50, starting from epoch 10. Note that for smaller models $E = 5$

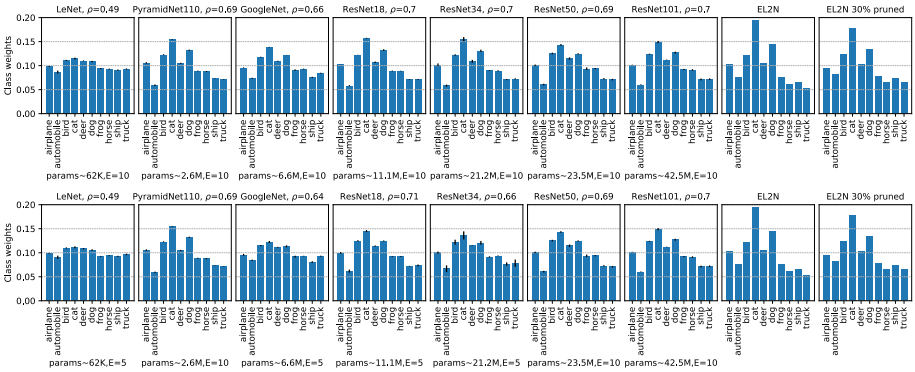

Fig. 8 Class weights implicitly learned by LeNet, PyramidNet110, GoogleNet, ResNet18, ResNet34, ResNet50 and ResNet101 on CIFAR10 using our method with $E = 10$ (top) and with the best epoch E (bottom) for 4 different initializations (error bars) and class weights induced from EL2N scores before and after pruning 30% of the samples with lowest scores. Captions on top give average Spearman rank correlation ρ of sample weights over the 4 initializations for the architectures, whereas those on bottom give the amount of parameters and the epoch E

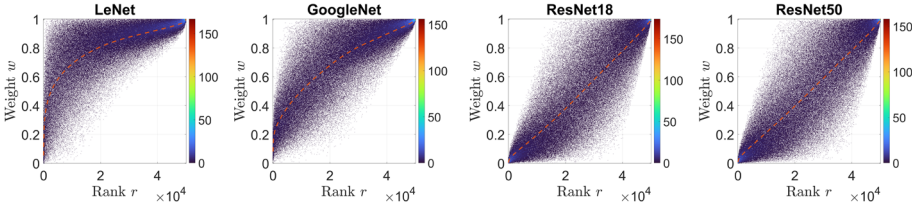

Fig. 9 Sample weights w_i over four different initializations for four different models trained on CIFAR10. Samples are sorted by their mean assigned weight $\bar{w}_i$. The x-axis is the resulting rank $r \in \{1, ..., M\}$ ($M = 50k$ for CIFAR10) from lowest $\bar{w}$ to highest $\bar{w}$. The dashed red line indicates $\bar{w}$. In case plotted points overlap, colour changes according to the number of overlapping points

performed slightly better than $E = 10$. The figure shows that using weights computed by LeNet [54] or GoogleNet [55] at epoch 5, instead of those computed by our method at epoch 10, when training a ResNet50 on CIFAR10 leads to worse results than using the weights computed by ResNet50. However, using weights computed by ResNet18 at epoch 5 leads to slightly better results. This further shows that sample weights are model-dependent. For ResNet18 we additionally report results with weights computed at $E = 10$, which suggests that weights that are worse for a given model are also worse when used for another model. Interestingly, for PyramidNet110 [56, 57] $E = 10$ leads to better results than $E = 5$, as for for the ResNets, suggesting that architectures using residual connections may learn similar sample importance distributions. Given these results, we hypothesize that PyramidNet110, ResNet18 and ResNet50 share some network similarities and therefore cannot be considered as completely different models, however additional experiments are required to verify this hypothesis as well as the possibility to use weights computed by a smaller ResNet into a bigger one.

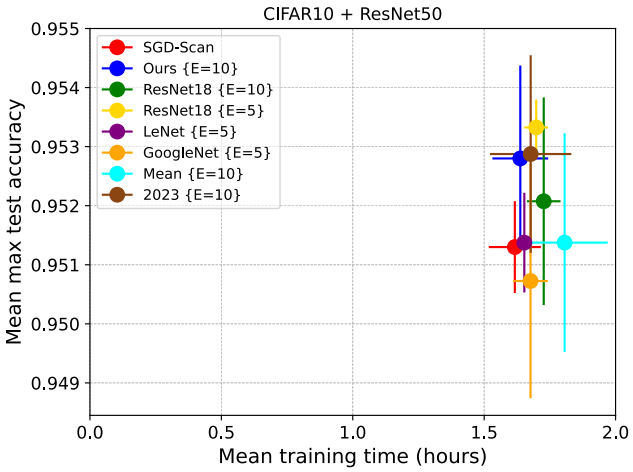

Fig. 10 Test accuracy of ResNet50 on CIFAR10 using our method with $E = 10$, but using other weights instead of computing them, as explained in the following. In both Figures, SGD-Scan and Ours E=10 are, respectively, the uniform sampling and our method with $E = 10$. *ResNet18* $E = 5$ and *ResNet18* $E = 10$, respectively, use the weights computed by ResNet18 with the same initialization at the specified epochs, *LeNet* uses weights computed by LeNet at epoch $E = 5$. *Mean* uses the average sample weights over the 4 runs computed with ResNet50, *2023* uses sample weights computed with ResNet50 when using seed 2023

4.2 Initialization dependency

Figure 9 shows initialization dependency on CIFAR10 for 4 runs of our model with different initializations each. For each run, the resulting weights $w_{i,j}$ are averaged $\bar{w}_i = \sum_{j=1}^{4} w_{i,j}$, where $j \in \{1, ..., 4\}$ indicates the runs. Samples are sorted in ascending order of $\bar{w}_i$ with $r(i)$ their resulting rank, i.e. the resulting new index $r(i) \in \{1, ..., M\}$. The plots show $(r(i), \bar{w}_{r(i)})$ as dashed red line and the dark blue dots are $(r(i), w_{r(i),j})$ with $j \in \{1, ..., 4\}$. So for each sample with index $r(i)$ we see the initialization dependence of weights $w_{r(i),\cdot}$ as vertical spread of the blue points. We observe that for all models the variation in $w_{r(i),\cdot}$ is small for the hardest samples with $r(i) \lesssim M$ and all assigned weights $w_{r(i),\cdot} \gtrsim 1 - \epsilon$, with some acceptable threshold ϵ, say $\epsilon = 0.1$. These samples are consistently detected as hard and could be identified reliably in a pre-processing step. The same is true for the easiest samples $r(i) \gtrsim 1$ and $w_{r(i),\cdot} \lesssim \epsilon$, but less pronounced for the small networks LeNet and GoogleNet. For medium hard samples, i.e. samples with $\bar{w}_{r(i)} \approx 0.5$ the variation of $w_{r(i),\cdot}$ is much higher. In many cases the same sample may be assigned $w_{r(i)} \ll 0.2$ in one initialization and $w_{r(i)} \gg 0.8$ in another initialization. Thus, depending on the initialization, a sample may be easy or hard for the same model.

Figure 11 shows the initialization dependency when training with 4 different seeds a ResNet50 on CIFAR100. Here we can see that samples are generally harder than CIFAR10. This is expected as there are 100 classes with 500 samples each, instead of 10 classes with 5000 samples each. Moreover, the plot confirms that hard samples are consistently detected as hard, whereas for medium and easy samples the variance is larger.

We hypothesize that in a dataset there may be multiple images conveying same learning information (i.e. similar features). Therefore, samples that come first in the training schedule are harder to learn than similar samples coming later in the training process. Consequently, the weight assigned to a sample depends on the order the samples are fed to the model.

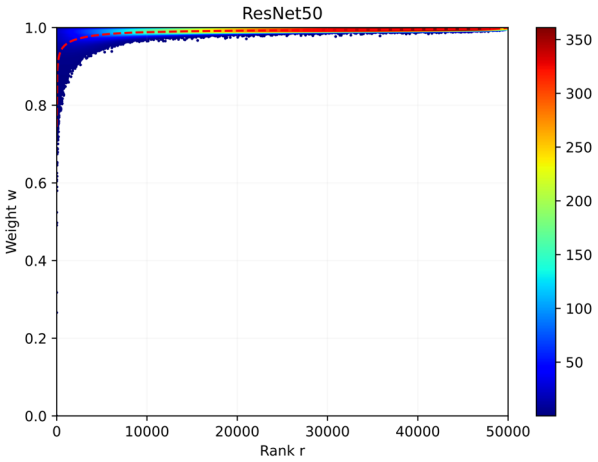

Fig. 11 Sample weights w_i over four different initializations for ResNet50 trained on CIFAR100. Samples are sorted by their mean assigned weight $\bar{w}_i$. The x-axis is the resulting rank $r \in \{1, ..., M\}$ ($M = 50k$ for CIFAR100) from lowest $\bar{w}$ to highest $\bar{w}$. The dashed red line indicates $\bar{w}$. In case plotted points overlap, colour changes according to the number of overlapping points

In Figure 10, we report results of two additional variants of our method, compared to uniform sampling and our approach with $E = 10$ on CIFAR10. Both variants start training a ResNet50 as our method until epoch $E = 10$, then update the sample weights as: the average sample weights (*Mean*) computed when using our approach with ResNet50 on 4 different runs, and sample weights computed in a specific run (with seed 2023). In other words, we still use 4 runs per method, but the weights are given as specified earlier. The figure shows that using the average sample weights leads to slightly worse results, whereas using sample weights computed by a different seed (except for the last run with seed 2023, which uses its original sample weights) can lead to comparable results. This suggests that there is a mild initialization dependency.

We conclude, that sample difficulty is initialization-dependent, when computed from model output $f_\theta(x_i)_{y_i}$. Consequently, it may not necessarily be possible to pre-compute (i) sample difficulty, (ii) an optimal selection of samples, or (iii) an optimal choice of per-sample weights. It may be necessary that weight assignment is done during training, as proposed here.

4.3 Task dependency

We show the task dependency by running our method on a subset of CIFAR10, where the third most confused class (*dog*) is removed. We denote this subset *CIFAR9*. Not surprisingly, Figure 12 shows that the class weights slightly change together with the task, although some general agreement on the class importance is retained. Of particular interest are the classes *bird* and *cat* in ResNet18, which in *CIFAR9* appear to be very close in difficulty, in contrast to the class weights in CIFAR10 (Figure 8), where it is evident that class *cat* is harder than *bird*.

Note that we adjust the training iterations for this experiment according to the data size (45000) and therefore run 70200 iterations. Here the learning rate is decayed after 21060, 42120, 56160 iterations, that is (60, 120, 160 epochs) $\times$ 351 iterations per epoch.

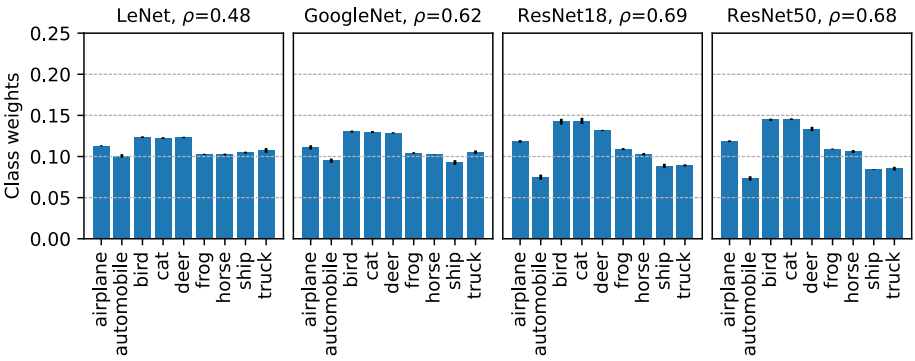

Fig. 12 Class weights implicitly learned by LeNet, GoogleNet, ResNet18 and ResNet50 using our method for 4 different initializations (error bars) on *CIFAR9*, a subset of CIFAR10 without class *dog*. Captions on top give average Spearman rank correlation ρ of sample weights over the 4 initializations for the architectures

4.4 Hypotheses and observations on sample importance

We hypothesize that:

- the later sample importance weights are computed (C), the more accurate they are;
- the sooner sample importance weights are applied (A), the greater their benefit.

The experimental analysis on CIFAR10 in Fig. 14 shows that the first hypothesis is not true, as results drop considerably when applying at epoch 10 sample weights computed at later epochs (purple line). This is further confirmed by the performance drop for later epochs in the blue line, representing sample weights computed and applied at the same epoch. On the other hand, the second hypothesis is confirmed by the green line, which shows that sample weights computed at epoch 10 are even better when applied earlier in training. This suggests that in order to get a performance boost, weights need to be applied as soon as possible. Note that applying at epoch 0 sample weights computed at epoch 10 requires a training restart, therefore it requires additional iterations, but it reaches slightly higher accuracy in the end. This might be due to the fact that the method “looks into the future" by training for some epochs, then restarts training by applying knowledge acquired by looking into the future. However this is not always true, as the light green line shows that weights computed at epoch 20 achieve comparable performance to those computed at epoch 10, when applied at epoch 10. These weights also lead to a slight improvement when applied at epoch 100, but in all other cases they lead to performance comparable to the baseline (red line).

Figure 13 shows that class importance weights (and therefore sample weights) evolve during training, from similarly difficult to similarly easy. Moreover, the figure shows how class weights in subsequent plots are correlated. Sample weights computed after epoch 100 are less correlated than weights computed earlier in training. This potentially suggests that updating the importance distribution every N epochs or in an online fashion could give even more benefits. In Sect. 5.2, we show that updating the importance distribution online does not lead to additional benefits, confirming that using estimates computed early enough is sufficient.

We conclude that sample importance weights computed and applied early enough can boost training performance of an image classifier when the given dataset has a difficulty imbalance.

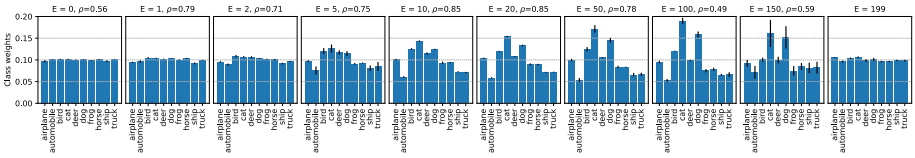

Fig. 13 Class weights implicitly learned at different epochs by ResNet50 on CIFAR10 using our method for 4 different initializations (error bars). Captions on top give the epoch at which weights are computed and the Spearman rank correlation ρ between the mean sample weights in a chart and the chart on its right (e.g. ρ between mean E=0 and E=1 weights on the first chart, E=1 and E=2 on the second, and so on)

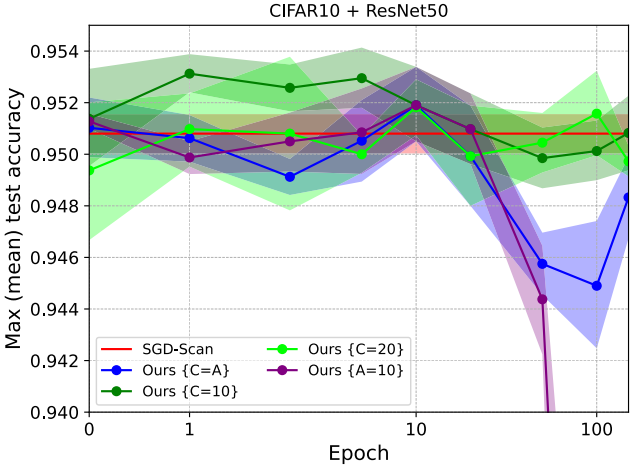

Fig. 14 Test accuracy of ResNet50 on CIFAR10 using our approach with sample weights computed (C) and/or applied (A) at the end of different epochs. Tested epochs are 0, 1, 2, 5, 10, 20, 50, 100 and 150

4.5 Choice of E in presence of class imbalance

We study the behaviour our method with variations of E in presence of imbalance. To this end, we extended the experiment from Fig. 14 to the artificially imbalanced CIFAR100 setup described in Sect. 3.2, and evaluated performance across different values of E. We present results in Fig. 16, which shows that our method outperforms SGD-Scan with peaks in the range of $E = 5$ to $E = 20$, and gradually declining thereafter. These results reinforce our core finding: as long as the bias is introduced early enough in training—that is, $E \leq 30$—our method consistently outperforms SGD-Scan, especially under significant class imbalance. This supports our broader claim that early emphasis on informative samples leads to more efficient learning.

Choice of E on a smaller subset of the data To further assess the robustness of our method to the choice of E, and to explore the potential for tuning E using less data, we repeated the experiment described in Sect. 4.5 on just 10% of the training set—using stratified sampling to preserve class imbalance (Fig. 17). Interestingly, the peak remains around $E = 20$ with consistent performance decay beyond that point, but the overall curve is less stable and values like $E = 5$ or $E = 10$ are no longer close in performance.

This suggests that identifying a good E value using only a small subset may be feasible, but less reliable, and tuning on limited data does not directly translate to the complete dataset.

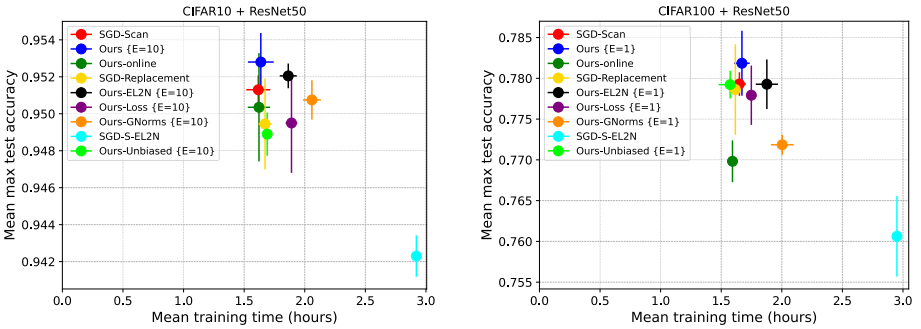

Fig. 15 Mean maximum test accuracy with respective standard deviation vs. mean compute time of ResNet50 on CIFAR10 (left) and CIFAR100 (right) for *SGD-Scan*, our method, and the following training algorithms: our method with sample importance distribution updated online (*Ours-online*), SGD with replacement (*SGD-Replacement*), our method with EL2N (without ensembling, *Ours-EL2N*), loss (*Ours-Loss*), gradient norms (*Ours-GNorms*) importance metrics, SGD sampled proportional to the provided EL2N scores (without pruning, *SGD-S-EL2N*), and our method when using unbiased gradient updates (*Ours-Unbiased*). Upper left is better. Ours is the only method that outperforms SGD at comparable runtime

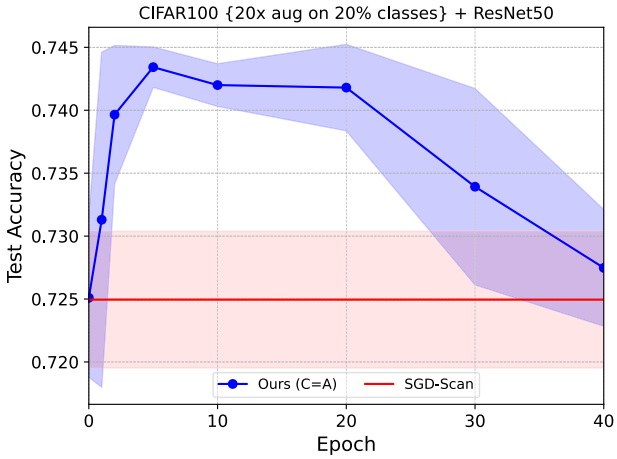

Fig. 16 Test accuracy of ResNet50 on CIFAR100 for standard training (*SGD-Scan*) and Ours, where 20% of classes are randomly selected and augmented 20 times. Shaded regions indicate standard deviation. Tested epochs are 0, 1, 2, 5, 10, 20, 30, 40

This is likely due to the fact that learning behaviours on a smaller dataset have inherently different gradient trajectories. With fewer examples, the model tends to converge faster and may overfit to class-specific patterns that are not representative of the full distribution. As a result, the point in training when updating the sampling distribution becomes most beneficial may occur earlier or later depending on the dataset size. Therefore, while small-scale tuning can offer a rough estimate, it may not yield optimal results for the full training regime.

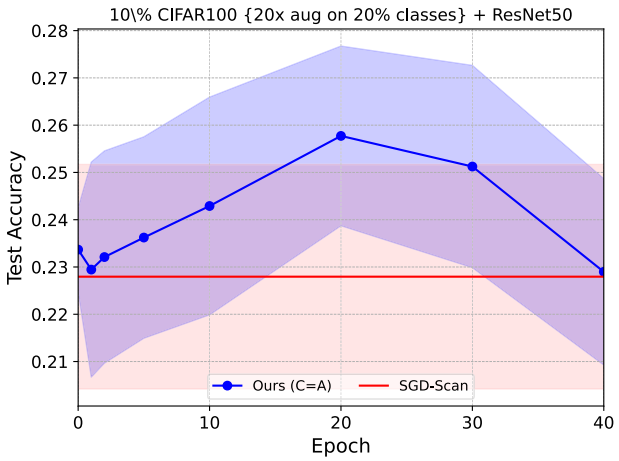

Fig. 17 Test accuracy of ResNet50 on a 10% stratified random subset of CIFAR100 for standard training (*SGD-Scan*) and Ours, where 20% of classes are randomly selected and augmented 20 times. Shaded regions indicate standard deviation. Tested epochs are 0, 1, 2, 5, 10, 20, 30, 40

5 Ablation studies

In this section, we report ablation studies further motivating our method choices. In particular, we report results for variants of our method using different importance metrics, a variant of our method computing weights online, a variant doing unbiased sampling, and a variant sampling proportional to EL2N scores. In all the evaluations, our method outperforms the variants.

5.1 Importance metric choice

In Figure 15, we report results for alternative importance metrics (loss, EL2N [8], gradient norms). The figure motivates our importance metric choice (4), as it is the best performing.

Note that to compute the gradient norms with our current framework we had to loop over each minibatch after computing the per-sample losses to compute the per-sample gradient. This procedure took approximately 35 min for the single epoch E. Doing this online would be prohibitive.

5.2 Computing weights online and SGD with replacement

To the best of our knowledge, we are the first to empirically show that it is possible to estimate sample importance once early in training and to achieve better results by continuing training using this estimate as sampling distribution.

We here investigate whether using our method in an online fashion would improve the performance. Additionally we compare our method against SGD with uniform sampling (i.e. with replacement).

In Fig. 15, on the other hand, we show that using our method in an online fashion (i.e. updating the sampling distribution after every iteration) leads to worse results. The figure also shows the results for SGD with replacement.

5.3 Biased vs unbiased sampling

We propose an SGD-based method biased towards important samples, as we assume that the given dataset is biased (i.e. it has an imbalance in sample importance). For comparison, in Fig. 15 we report results for an unbiased version of our method, where we weight the gradient steps by the inverse of our sample weights (4). The figure shows that using a biased version of our method leads to better results.

5.4 Sampling proportional to EL2N scores

Lastly, Fig. 15 shows the results for a method that samples proportional to the provided EL2N scores (instead of pruning and using SGD-Scan on the pruned data). The experiment shows that EL2N scores used as sample weights do not perform as well as the same scores used for data pruning and then training using SGD-Scan on the resulting pruned dataset.

6 Related work

Our work relates to Focal Loss [25] in the use of focus and sample difficulty concepts, and to the data pruning method of Paul et al. [8], which also aims to estimate sample importance once only early in training. However, our method differs conceptually: we downsample less important samples instead of downweighting their loss, and we compute sample importance only once without training a new model from scratch.

Other data pruning works suggest focusing on high influence and high memorization training samples [4, 5]. Jiang et al. [6] rank samples according to their structural regularities, while Toneva et al. [3] suggest a forgetting score of the number of forgetting events a sample underwent during training. All of these methods suffer from the drawback that they have to compute sample importance first, prune accordingly and then train a new model from scratch, requiring additional training iterations, which our method avoids. Moreover, this overhead increases considering that scores are likely model-specific [3, 6, 9], which implies that sample scoring needs to be repeated for every model and every dataset.

Importance sampling shows that it is possible to speed up training by sampling proportional to the gradient norms [13–15], while producing an umbiased estimator with convergence guarantees. In practice, these approaches are currently computationally prohibitive as they require computing the full gradient for each sample individually, implying a large amount of memory load and computational overhead. To overcome this issue, several approximations were proposed, including methods which sample proportional to the norm of the gradient computed with respect to the last layer [16, 19, 20] or to the loss [17, 18, 21].

A different approach is adopted by *hard example mining* methods, which try to boost a model learning by biasing it towards hard samples, under the assumption that they are more informative for generalization. Some methods achieve this by sampling hard samples more often [22, 23, 26] or giving their loss a higher weight [25], whereas others by backpropagating a subset of hard samples in a batch after every forward pass [11, 24, 27–30]. Similarly, *coresets selection* methods [36–38] try to find sample subsets during training approximating the gradient of the full training set. Lastly, some methods determine a sequence of batches by solving an optimization problem prior training [12] or produce a *curriculum* for training, originally presenting samples from easy to hard [31], then extended to more complex strategies [10, 32–34].

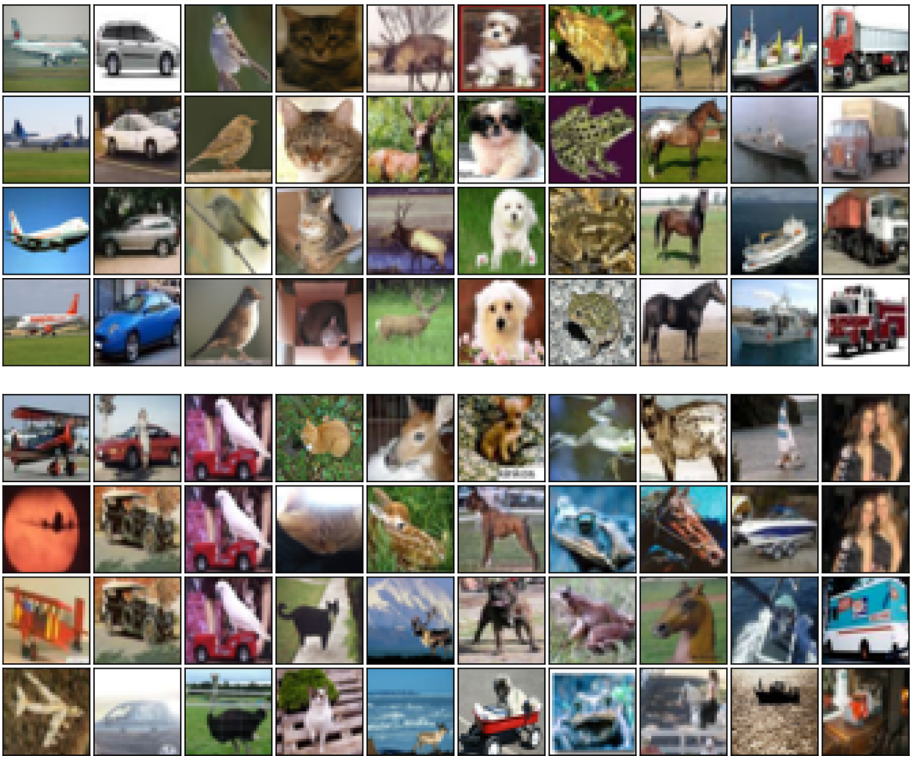

Fig. 18 Sample with the easiest (top) and highest (bottom) weight per class (columns) for 4 different initializations (rows) of ResNet50 on CIFAR10 with $E = 10$

Despite being promising, these methods are often less effective in practice as they introduce additional computational overhead to compute sample importance. In fact, gradient-based *importance sampling* [13–15] methods require to compute per-sample gradients, which is currently computationally unfeasible [20], whereas other methods require training additional networks [11, 21, 32, 35], computing sample importance before training starts [10, 12, 30], running expensive computations during training [29, 33, 34] or solving bi-level optimization problems [35–38], based on conservative estimates [8]. In constrast, we compute sample importance *only once early in training* using (4) and then we bias the model towards important samples by sampling proportionally to it for the remaining iterations. By doing so, we emphasize important samples and improve learning performance, avoiding the different forms of overheads required by the other discussed methods.

7 Visual inspection

In Fig. 18, we visualize the easiest (top) and hardest (bottom) sample per class (columns) and per seed (rows). These figures reveal different interesting observations:

- often the difference between easier and harder is the different orientation of the object inside the image (e.g. the frontal cat is considered easy, whereas the cat seen from other angles is scored as being harder, perhaps because there are less images like so)

- our method succeeds in identifying easy and hard samples: comparing the easy samples in Fig. 18 (top) with the hard samples in Fig. 18 (bottom), we see a clear visual difference in whether the image content is prototypical of the class label. Consider e.g. horse class (third column from the right) where the top column shows entire horses from the side, whereas the lower column shows partial and slightly occluded views.
- some of the hardest samples are outliers (e.g. the first 2 images from top of the truck column (right-most column in Fig. 18, bottom) where the images mainly shows people.

8 Discussion and conclusion

We introduce Focal Sampling, a method that biases SGD towards samples that are found to be more important using vanilla SGD for a few training epochs.

In contrast to state-of-the-art, our method does not require any additional training iterations or overhead to estimate sample importance.

We find that our method is the only one to converge faster than standard training on CIFAR10, CIFAR100, and iNaturalist2021 to the same maximum mean accuracy, and to achieve higher maximum mean accuracy when given the same time as SGD-Scan. On the more balanced ImageNet-1K it achieves comparable results.

These results suggest that it is possible to define an importance distribution over the training samples in a few epochs and that using this distribution as sampling distribution for the remaining iterations improves results with respect to standard training.

Moreover, we find that our method makes efficient use of multiple augmentations, as it can select important samples from a larger pool of images, allowing the model to avoid easy augmented samples and learn from challenging ones. This allows our method to achieve improved performance with approximately the same training time as needed to train without augmentations. We see interesting opportunities to leverage our method also in applications where there is lack of data and where multiple augmentations could be key to achieve better performances. Here, our method allows the automatic focusing to those augmentations that actually benefit learning.

We observe that our method works best when the given dataset contains a difficulty imbalance and improve results on the much bigger iNaturalist2021, without requiring to tune our hyperparameters. Despite this, dependence on the number of pretraining epochs E and its correlation with datasets and tasks remains to be explored. Assuming that the parameters can be re-used, extrapolated (e.g. as a percentage of the training epochs) or found in a fast manner, and considering that our method outperforms the baseline methods in all experiments within the time budget of SGD-Scan, it might be preferable to use our method on big imbalanced datasets requiring longer training times. In cases where re-using our settings is not possible, it may be required to tune the parameter E, making baselines without explicit parameters appear more advantageous (SGD-Prop or SGD-Rank, [17]). However, in our results these methods not only take longer, but also achieve lower accuracy. In addition, early stopping strategies may be adopted to tune our parameter E in an efficient way, while still preserving the method advantages.

We offer empirical studies on *class importance* suggesting that unlike commonly assumed, balancing classes by number of samples is not necessarily ideal. Our results instead show that a model trained with classes sampled according to *class importance* weights derived by our *sample importance* weights can outperform a model trained with balanced class weights.

Still, finer-grained importance weights (i.e. per sample, 4) are preferable as they allow the sampler to oversample the most important samples, independently from their class.

Lastly, our empirical analysis suggests that sample importance has mild model, initialization and task dependence, and that the sooner importance weights are applied, the better for the model convergence speed and accuracy.

Given our promising results, we hope to encourage further research on sample and class importance, to further understand the learning dynamics and optimize the training procedure.

9 Implementation and hardware details

We implement our code using PyTorch [58] and PyTorch Lightning [59] as Deep Learning frameworks and Hydra [60] to manage the configuration files. We take inspiration on how to combine these frameworks from existing templates[3,4].

If not stated differently, we run experiments on a single NVIDIA A100-SXM4-40GB. We use approximately 5500 GPU hours for the whole project. Experiments shown here took approximately 700 GPU hours to compute.

Appendix A: Hyperparameter tuning

a) Sensitivity of E: Figure 19 shows that for $\gamma = 0.5$ our method is slighlty sensitive to the number of pretraining epochs E with the uniform random sampler. For CIFAR10 we chose $E = 10$, but values close to 10 and between 5 and 20 can also be good candidates, whereas for CIFAR100 only $E = 1$ leads to better results than the baseline, and other values lead to comparable or slightly worse performance.

b) Sensitivity of γ: In Fig. 20, we report the sensitivity analysis of γ for the chosen values of E for both CIFAR10 ($E = 10$) and CIFAR100 ($E = 1$). The figure shows that multiple values of γ can be used for the same value of E to achieve comparable results. For instance, in CIFAR10 both $\gamma = 0.5$ and $\gamma = 1$ achieve better performance than the baseline, whereas only $\gamma >= 2$ lead to drop in performance. For CIFAR100, on the other hand, multiple values of γ lead to better performance than the baseline (0.1, 0.5, 1.5, 3, whereas the other tested values lead to comparable or slightly worse results.

Lastly, Fig. 20 shows that $\gamma = 0.5$ leads to the best performance for both datasets, and therefore we use this for all experiments.

[3] https://github.com/ashleve/lightning-hydra-template.

[4] https://github.com/grok-ai/nn-template.

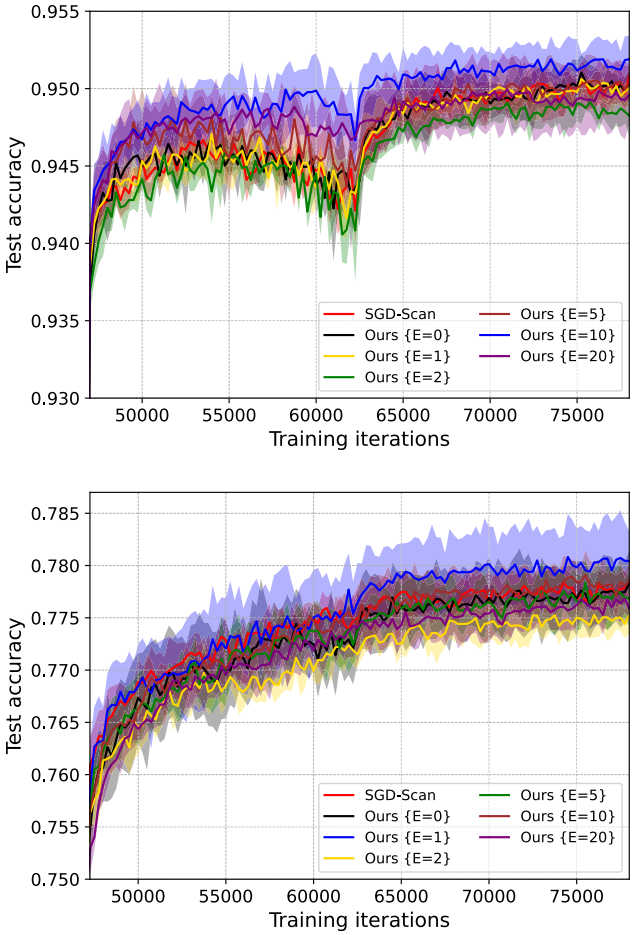

Fig. 19 Sensitivity of E when using our method with ResNet50 on CIFAR10 (top) and CIFAR100 (bottom) with $\gamma = 0.5$. The blue lines correspond to the results shown in the main paper

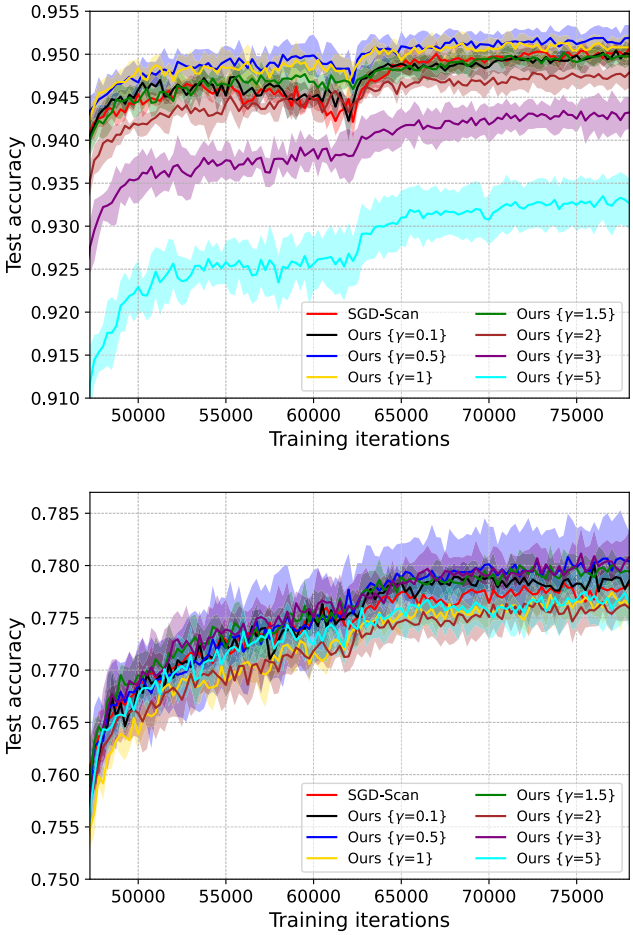

Fig. 20 Sensitivity of γ when using our method with ResNet50 on CIFAR10 with $E = 10$ (top) and CIFAR100 with $E = 1$ (bottom). The blue lines correspond to the results shown in the main paper

Acknowledgements This work was funded by the Helmholtz School for Data Science in Life, Earth, and Energy (HDS-LEE) and supported by the Helmholtz Association's Initiative and Networking Fund on the HAICORE@FZJ partition [61]. The authors gratefully acknowledge the computing time granted by the JARA Vergabegremium and provided on the JARA Partition part of the supercomputer JURECA [62] at Forschungszentrum Jülich. The authors kindly thank Mansheej Paul for providing the EL2N scores [8].

Author Contributions Alessio Quercia wrote the main manuscript text and the code for the experiments. Fernanda Nader Nieto contributed to the third paper revision and to Section IV.E of the main paper. Hanno Scharr prepared Figure 9. Abigail Morrison, Hanno Scharr and Ira Assent contributed to the supervision of the project. All authors reviewed/edited the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval The main effect of our work is to speed up the learning process and automatically balance classes. Speedup reduces energy consumption and potentially allows for using less powerful hardware for training, a tiny step towards democratizing AI. Balancing classes may help to emphasize underrepresented samples and thus may raise visibility of minorities in data. It depends on the target application if this results in highly desired fair treatment of minorities or an unfair deviation from underlying distributions.

Reproducibility Statement All presented results are reproducible using our code, which is available as an anonymous repository at https://jugit.fz-juelich.de/ias-8/sgd_biased.

References

1. Schuhmann C, Beaumont R, Vencu R, Gordon CW, Wightman R, Cherti M, Coombes T, Katta A, Mullis C, Wortsman M, Schramowski P, Kundurthy SR, Crowson K, Schmidt L, Kaczmarczyk R, Jitsev J (2022) LAION-5b: an open large-scale dataset for training next generation image-text models, In: Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track. [Online]. Available: https://openreview.net/forum?id=M3Y74vmsMcY
2. Cherti M, Beaumont R, Wightman R, Wortsman M, Ilharco G, Gordon C, Schuhmann C, Schmidt L, Jitsev J (2022) Reproducible scaling laws for contrastive language-image learning. [Online]. Available: https://arxiv.org/abs/2212.07143
3. Toneva M, Sordoni A, Combes R T d, Trischler A, Bengio Y, Gordon GJ (2018) An empirical study of example forgetting during deep neural network learning, arXiv:1812.05159
4. Feldman V, Zhang C (2020) What neural networks memorize and why: discovering the long tail via influence estimation. Adv Neural Inf Process Syst 33:2881–2891
5. Feldman V (2020) Does learning require memorization? a short tale about a long tail, In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, pp. 954–959
6. Jiang Z, Zhang C, Talwar K, Mozer MC (2020) Characterizing structural regularities of labeled data in overparameterized models, arXiv:2002.03206
7. Pruthi G, Liu F, Kale S, Sundararajan M (2020) Estimating training data influence by tracing gradient descent. Adv Neural Inf Process Syst 33:19 920-19 930
8. Paul M, Ganguli S, Dziugaite GK (2021) Deep learning on a data diet: finding important examples early in training, *Advances in Neural Information Processing Systems*, 34
9. Sorscher B, Geirhos R, Shekhar S, Ganguli S, Morcos AS (2022) Beyond neural scaling laws: beating power law scaling via data pruning, arXiv:2206.14486
10. Hacohen G, Weinshall D (2019) On the power of curriculum learning in training deep networks, In: International Conference on Machine Learning. PMLR, pp. 2535–2544
11. Mindermann S, Brauner JM, Razzak MT, Sharma M, Kirsch A, Xu W, Höltgen B, Gomez AN, Morisot A, Farquhar S et al. (2022) Prioritized training on points that are learnable, worth learning, and not yet learnt, In: International Conference on Machine Learning. PMLR, pp. 15630–15649
12. Banerjee S, Chakraborty S (2021) Deterministic mini-batch sequencing for training deep neural networks. Proceed AAAI Conf Artif Intell 35(8):6723–6731
13. Needell D, Ward R, Srebro N (2014) Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm, Advances in neural information processing systems, 27
14. Alain G, Lamb A, Sankar C, Courville A, Bengio Y (2015) Variance reduction in SGD by distributed importance sampling, arXiv:1511.06481

15. El Hanchi A, Stephens D, Maddison C (2022) Stochastic reweighted gradient descent, In: International Conference on Machine Learning. PMLR, pp. 8359–8374
16. Zhao P, Zhang T (2015) Stochastic optimization with importance sampling for regularized loss minimization. In: international conference on machine learning. PMLR, pp. 1–9
17. Schaul T, Quan J, Antonoglou I, Silver D (2015) Prioritized experience replay, arXiv:1511.05952
18. Katharopoulos A, Fleuret F (2017) Biased importance sampling for deep neural network training, arXiv:1706.00043
19. Johnson TB, Guestrin C (2018) Training deep models faster with robust, approximate importance sampling, Advances in Neural Information Processing Systems, 31
20. Katharopoulos A, Fleuret F (2018) Not all samples are created equal: Deep learning with importance sampling, In: International conference on machine learning. PMLR, pp. 2525–2534
21. Ganapathiraman V, Rodriguez FC, Joshi A (2022) Impon: efficient importance sampling with online regression for rapid neural network training
22. Loshchilov I, Hutter F (2015) Online batch selection for faster training of neural networks, arXiv:1511.06343
23. Simpson AJ (2015) oddball sgd: Novelty driven stochastic gradient descent for training deep neural networks, arXiv:1509.05765
24. Shrivastava A, Gupta A, Girshick R (2016) Training region-based object detectors with online hard example mining, In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 761–769
25. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection In: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988
26. Chang H-S, Learned-Miller E, McCallum A (2017) Active bias: training more accurate neural networks by emphasizing high variance samples, Advances in Neural Information Processing Systems, 30
27. Jiang A H, Wong D L-K, Zhou G, Andersen D G, Dean J, Ganger G R, Joshi G, Kaminksy M, Kozuch M, Lipton ZC, et al. (2019) Accelerating deep learning by focusing on the biggest losers, arXiv:1910.00762
28. Kawaguchi K, Lu H (2020) Ordered sgd: a new stochastic optimization framework for empirical risk minimization, In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 669–679
29. Dong C, Jin X, Gao W, Wang Y, Zhang H, Wu X, Yang J, Liu X (2021) One backward from ten forward, subsampling for large-scale deep learning arXiv:2104.13114
30. Lu Y S, Zamoshchin D, Chen Z (2022) Diet selective-backprop: Accelerating training in deep learning by pruning examples, Stanford University, Tech. Rep. [Online]. Available: http://cs231n.stanford.edu/reports/2022/pdfs/93p.pdf
31. Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning, In: Proceedings of the 26th annual international conference on machine learning, pp. 41–48
32. Jiang L, Zhou Z, Leung T, Li L-J, Fei-Fei L (2018) Mentornet: learning data-driven curriculum for very deep neural networks on corrupted labels, In: International conference on machine learning. PMLR, pp. 2304–2313
33. Zhou T, Wang S, Bilmes J (2020) Curriculum learning by dynamic instance hardness. Adv Neural Inf Process Syst 33:8602–8613
34. Zhou T, Wang S, Bilmes J (2021) Curriculum learning by optimizing learning dynamics, In: International conference on artificial intelligence and statistics. PMLR, pp. 433–441
35. Coleman C, Yeh C, Mussmann S, Mirzasoleiman B, Bailis P, Liang P, Leskovec J, Zaharia M (2019) Selection via proxy: efficient data selection for deep learning, arXiv:1906.11829
36. Mirzasoleiman B, Bilmes J, Leskovec J (2020) Coresets for data-efficient training of machine learning models, In: International Conference on Machine Learning. PMLR, pp. 6950–6960
37. Killamsetty K, Sivasubramanian D, Ramakrishnan G, Iyer R (2020) Glister: generalization based data subset selection for efficient and robust learning, arXiv:2012.10630
38. Killamsetty K, Durga S, Ramakrishnan G, De A, Iyer R (2021) Grad-match: gradient matching based data subset selection for efficient deep model training, In: International Conference on Machine Learning. PMLR, pp. 5464–5474
39. Quercia A, Morrison A, Scharr H, Assent I (2023) Sgd biased towards early important samples for efficient training, In: 2023 IEEE International Conference on Data Mining (ICDM). IEEE
40. Fort S, Dziugaite GK, Paul M, Kharaghani S, Roy DM, Ganguli S (2020) Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. Adv Neural Inf Process Syst 33:5850–5861
41. Arpit D, Jastrzkebski S, Ballas N, Krueger D, Bengio E, Kanwal MS, Maharaj T, Fischer A, Courville A, Bengio Y et al. (2017) A closer look at memorization in deep networks, In: International conference on machine learning. PMLR, pp. 233–242

42. Mangalam K, Prabhu VU (2019) Do deep neural networks learn shallow learnable examples first? in: ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena. [Online]. Available: https://openreview.net/forum?id=HkxHv4rn24
43. Castells T, Weinzaepfel P, Revaud J (2020) Superloss: a generic loss for robust curriculum learning. Adv Neural Inf Process Syst 33:4308–4319
44. AdeelH (2020) Pytorch-multi-class-focal-loss, https://github.com/AdeelH/pytorch-multi-class-focal-loss/blob/master/focal_loss.py
45. Mukhoti J, Kulharia V, Sanyal A, Golodetz S, Torr P, Dokania P (2020) Calibrating deep neural networks using focal loss. Adv Neural Inf Process Syst 33:15 288-15 299
46. Krizhevsky A, Hinton G. et al. (2009) Learning multiple layers of features from tiny images
47. Van Horn G, Cole E, Beery S, Wilber K, Belongie S, Mac Aodha O (2021) Benchmarking representation learning for natural world image collections, In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 12884–12893
48. Wightman R, Touvron H, Jégou H (2021) Resnet strikes back: an improved training procedure in timm, arXiv:2110.00476
49. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vision 115(3):211–252
50. Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV (2019) Autoaugment: learning augmentation strategies from data, In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 113–123
51. Müller SG, Hutter F (2021) Trivialaugment: tuning-free yet state-of-the-art data augmentation, In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 774–782
52. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
53. Cubuk ED, Zoph B, Shlens J, Le QV (2020) Randaugment: practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp. 702–703
54. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
55. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9
56. Han D, Kim J, Kim J (2017) Deep pyramidal residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5927–5935
57. Han D, Kim J, Kim J (2017) Deep pyramidal residual networks, IEEE CVPR
58. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in pytorch
59. Falcon W and The PyTorch Lightning team (2019) PyTorch Lightning, 3. [Online]. Available: https://github.com/PyTorchLightning/pytorch-lightning
60. Yadan O (2019) Hydra - a framework for elegantly configuring complex applications, Github. [Online]. Available: https://github.com/facebookresearch/hydra
61. Kesselheim S, Herten A, Krajsek K, Ebert J, Jitsev J, Cherti M, Langguth M, Gong B, Stadtler S, Mozaffari A et al. (2021) Juwels booster–a supercomputer for large-scale ai research, In: International conference on high performance computing. Springer, pp. 453–468
62. Thörnig P (2021) Jureca: data centric and booster modules implementing the modular supercomputing architecture at jülich supercomputing centre. J Large-scale Res Facil JLSRF 7:A182–A182

Alessio Quercia received his Bachelor's degree in Computer Science from Sapienza University of Rome and his Master's degree in Computer Science from the University of Milan in 2020. He was a Research Intern at IBM Research Zurich from 2019 to 2020 and a Research Fellow at Sapienza University of Rome from 2020 to 2021. Currently, he is waiting to defend his Ph.D. in Computer Science at RWTH Aachen University and doing his post-doc at IAS-8, Forschungszentrum Jülich, where he focuses on Data Efficiency and Transfer Learning and MTL.

Fernanda Nader received her B.Sc. in Physics from the Universidad Nacional de Colombia and her M.Sc. in Data Science from RWTH Aachen University in 2024. She previously worked as a researcher in geophysics at LMU Munich and the Colombian Geological Survey, and she is currently a Research Associate at IAS-8, Forschungszentrum Jülich.

Abigail Morrison is a computational neuroscientist at the Research Centre Jülich and the RWTH Aachen. She received her MSc in artificial intelligence in 2001 from Edinburgh University and her PhD in computational neuroscience in 2006 from the University of Freiburg. Between 2006 and 2009, she was a scientific researcher at the RIKEN Brain Science Institute in Japan; she subsequently held a junior professorship at the University of Freiburg from 2009 to 2012, as well as a group leadership at the Bernstein Center Freiburg, followed by a professorship at Ruhr University of Bochum from 2012 to 2020. Currently, she is Professor of Computational Neuroscience at the Department of Computer Science, RWTH Aachen and Group Leader of "Computation in Neural Circuits" at the Institute for Advanced Simulation, IAS-6: Computational and Systems Neuroscience, Forschungszentrum Juelich. Her primary neuroscientific research focus is learning, representation and computation in spiking neural networks. Her technological research interests include neuroinspired computation and developing high-performance simulation technology for spiking neural networks, in particular the NEST simulator (www.nest-simulator.org) and its modeling language toolchain, NESTML.

Hanno Scharr received his Diploma degree in Physics from Heidelberg University, Germany in 1996. He continued in Heidelberg with a graduate scholarship at the Center of Excellence "Modeling and Scientific Computing in Mathematics and Natural Sciences" and received his PhD in 2000 also from Heidelberg University in Physics for his work on "Optimal Operators in Digital Image Processing". In 2014, he was granted a habilitation in computer science from Frankfurt University. After a first postdoc in Heidelberg, he joined Intel Research in Santa Clara, California, in 2002 for a year where he worked on real-time denoising algorithms for nano-machining tools. In 2003, he moved to Forschungszentrum Jülich, Germany, leading a junior group on "Automatic Imaging and Quantitative Image Processing in Environmental Plant Sciences" with a young investigator award. Subsequently, he led the Image Analysis Group of the Institute of Bio- and Geosystems: IBG-2 Plant Sciences at Forschungszentrum Jülich. Since 2021, he heads the Institute of Advanced Simulation (IAS-8): Data Analytics and Machine Learning at Forschungzentrum Jülich. Hanno authored and co-authored 150+ publications and patents, regularly serves as a reviewer for all major computer vision journals, and for 35+ international conference program committees. His main interests are quantitative image processing, computer vision, and machine learning with applications to natural sciences.

Ira Assent is full professor of computer science at Aarhus University, Denmark. She is a (part-time) director of the Institute for Advanced Simulation, Data Analytics, and Machine Learning (IAS-8) at the Forschungszentrum Juelich, Germany. She received the PhD degree from RWTH Aachen University, Germany. Her research interests include explainable AI, unsupervised and supervised learning, efficient and scalable algorithms for data analysis, and data management.