Machine Learning (2025) 114:275
https://doi.org/10.1007/s10994-025-06907-w

An efficient model training framework for green AI

Francesco Scala[1,2] · **Sergio Flesca**[2] · **Luigi Pontieri**[1]

Received: 3 April 2025 / Revised: 8 September 2025 / Accepted: 23 September 2025

Abstract
Training deep neural networks (DNNs) is increasingly recognized as a major contributor to the energy footprint of Artificial Intelligence (AI). Existing dataset pruning (DP) and active learning (AL) techniques reduce training data volumes but often introduce costly computations that undermine their energy-saving potential. This paper introduces *Play it Straight* and its enhanced variant *Re-Play it Straight*, two adaptive training algorithms that combine random subset sampling with lightweight AL-inspired instance selection. The proposed framework achieves a better balance between accuracy and energy efficiency by incrementally fine-tuning models on small, informative subsets, while controlling computational overhead. Experiments on multiple benchmark datasets demonstrate substantial reductions in training energy compared to state-of-the-art DP and AL methods, with *Re-Play it Straight* consistently delivering superior performance. These results highlight the potential of our approach to support more sustainable deep learning practices, contributing to the broader goals of Green AI.

Keywords Data pruning · Green-AI · Active learning · Energy efficiency · Sustainability

Editors: Riccardo Guidotti, Anna Monreale, Dino Pedreschi.

✉ Sergio Flesca
sergio.flesca@unical.it

Francesco Scala
francesco.scala@icar.cnr.it

Luigi Pontieri
luigi.pontieri@icar.cnr.it

[1] Institute of High Performance Computing and Networking (ICAR-CNR), Via P. Bucci, 87036 Rende (CS), Italy

[2] Department of Computer Engineering, Modeling, Electronics, and Systems Engineering (DIMES), University of Calabria, 87036 Rende (CS), Italy

Published online: 11 November 2025

1 Introduction

Recent advances in Artificial Intelligence (AI) have profoundly transformed a wide range of industries, offering innovative solutions to complex problems. Its applications span healthcare, finance, manufacturing and beyond, reshaping both professional and everyday life. However, this rapid diffusion raises serious concerns regarding the sharp increase in energy consumption and the consequent growth in carbon emissions (de Vries, 2023). A primary contributor to this issue is the training of Deep Learning models, which requires massive computational resources and large-scale datasets to develop effective Deep Neural Networks (DNNs), thereby significantly amplifying energy demand (Strubell et al., 2019). In particular, the electricity consumed during model training is a key driver of AI-related carbon emissions. Since electricity generation still largely relies on non-renewable sources such as coal and natural gas, that is the backbone of global energy supply in various countries, for instance in Italy (Flesca et al., 2022), the training of AI models further aggravates the challenges of climate change.

As a response to these environmental challenges, the emerging research field of Green AI is devoted to reducing the energy and carbon footprint of AI systems while promoting the development of energy-conscious models and algorithms. Key areas of Green-AI research include: (i) *Minimizing energy usage*: minimizing energy consumption by designing training and inference methods that are less resource-intensive; (ii) *Harnessing renewable energy sources*: leveraging renewable energy sources, such as solar and wind power, to sustain AI workflows; and (iii) *Optimizing hardware*: optimizing hardware architectures specifically tailored for efficient AI computations.

Within this context, our work aims at minimizing energy consumption by focusing on the efficient integration of data selection strategies and deep learning methods, with the goal of mitigating the energy impact of training large-scale DNN models while preserving high levels of predictive accuracy.

1.1 Existing solutions, open issues and research gap

Several approaches have been proposed to reducing training costs by leveraging data selection/sampling methods. In this context, Data Pruning (DP) approaches (Guo et al., 2022) specifically attempt to extract a small subset (a.k.a. *coreset*) of a given dataset that can be used surrogate of the original dataset in performing a target learning task (Sachdeva & McAuley, 2023).

A wide variety of DP solutions have been proposed in the literature, which feature different strengths and weaknesses, and can be grouped in the following main categories: *geometry-based* methods (Sener & Savarese, 2017), *loss/error based methods* (Paul et al., 2021), *gradient matching methods* (Mirzasoleiman et al., 2020), *bilevel-optimization methods* (Yang et al., 2024), and *sub-modular methods* (Iyer et al., 2021). Unfortunately, most DP methods entail heavy computations, which may vanish the benefit of shrinking the training set in some application settings.

In fact, recent studies (Okanovic et al., 2024; Guo et al., 2022; Ayed & Hayou, 2023) revealed that random sampling schemes are a strong data-reduction baseline, which often achieves similar/superior time-to-accuracy performances to DP methods —presumably because of detrimental distribution-shift effects introduced by their underlying biased data

extraction strategies (Ayed & Hayou, 2023). Starting from this observation, the *Repeated Sampling of Random Subsets* (*RS2*) (Okanovic et al., 2024) method was recently proposed that attempts to cut training costs by randomly selecting a subset of data for each training epoch.

However, using *RS2* to reduce the overall training costs requires predetermining the number of data instances necessary to reach the target predictive accuracy. If this number is under-estimated, and one still wants to achieve the desired accuracy level, the entire procedure must be redone, so incurring redundant training, increased computational overhead, and unnecessary energy consumption.

In principle, as suggested in Park et al. (2022); Scala et al. (2024, 2025), Active Learning (AL) offers a potential remedy to the limitation of *RS2*, since it does not require predefining the data budget in advance. By iteratively selecting only the most informative instances, AL can reduce the training cost of DNN models over large labeled datasets, effectively focusing the learning process on compact yet highly representative subsets. Nonetheless, a mere application of conventional AL schemes relying on repeated full retraining over progressively larger data subsets, as proposed in Park et al. (2022), is likely to introduce significant computational overhead and energy consumption, a drawback that has been confirmed empirically in our experimentation.

It is also important to note that even computationally inexpensive AL strategies, such as uncertainty-based methods (e.g. least-confidence, entropy or margin sampling), may in some cases yield results that are less energy-efficient than standard full training. This counterintuitive outcome arises from the bias introduced in the instance selection process: when the selected samples are not sufficiently representative of the overall data distribution, the training dynamics can suffer from slow convergence or, in the worst case, from overfitting. Indeed, every AL cycle may cause a distribution shift in the training data and yield biased or noisy (i.e. high-variance) gradient estimates, owing to the discarded examples, as discussed in Yang et al. (2025). Since this may lead to low learning/convergence speed, the number of optimization steps required to achieve a given accuracy target may increase substantially, ultimately leading to higher energy consumption than conventional training approaches, as shown in Sect. 4.2.1.

1.1.1 Research gap

As pointed out in Okanovic et al. (2024), despite the surprisingly impressive performance of *RS2* in the light of its above-mentioned limitations, it is natural to regard it as a (strong) baseline for validating novel solutions addressing the problem of minimizing the computational costs needed to train a model with a given target accuracy. Conducting more research on devising more effective importance sampling approaches was suggested in Okanovic et al. (2024) as a promising way to further close the gap between *RS2* and training on the full dataset. Despite the solid accuracy performances of AL approaches to traditional label-scarce supervised learning settings and the little computational burden and convergence properties (Haimovich et al., 2024) of common loss and uncertainty-based data selection methods, very few studies concerning the problem of efficiently training DNN models have explored the opportunity to address it by repeatedly applying these methods in an AL-like fashion (Park et al., 2022) (analogously to the repeated random sampling procedure of *RS2*), instead of just using them to preliminary extract a fixed coreset for training the model. This

research gap presumably descends from the fact that the pure AL-based approaches to the problem explored so far have exhibited consistently underperformed the *RS2* baseline in extensive comparative studies (Okanovic et al., 2024). This behavior presumably descends from the fact that the distribution shifts of multiple AL cycles may lead to slower convergence of the training loss and/or towards a "false" optimum of it (i.e., a parameter setting that minimizes the loss w.r.t. the distribution of the training data but not w.r.t. that of the dataset) (Yang et al., 2025).
Thus, the following intriguing, yet unaddressed, research question arises from our analysis of the state of the art in the field:

RQ1) *Can AL approaches to efficient DNN training be improved in a way that allow them to close the gap (and hopefully beat) the strongRS2 baseline in cost-to-accuracy performances?*

1.2 Contribution

Motivated by the limitations of existing data pruning and sampling strategies for efficient DNN training and the above mentioned research gap, this paper introduces a novel framework that encompasses *Play it Straight* (originally proposed in Scala et al. (2025)) and its enhanced variant, *Re-Play it Straight*. Both algorithms are grounded in the central idea of synergistically integrating an *RS2*-based warm-up phase with an iterative, AL-inspired refinement process. This combination enables the model to be progressively improved through the inclusion of small, highly informative subsets of training instances, thereby achieving a favorable balance between computational efficiency and predictive performance. In general, AL methods rely on repeatedly choosing a subset of unlabeled data to retrain a model, till a desired accuracy level is reached or a predefined (labeling) budget is consumed. Such a basic iterative data-driven refinement scheme is here extended in several technical facets to address the problem of efficiently and effectively training a DNN model against a large collection of labeled data.

1.2.1 Overview of the proposed approach

As illustrated in Fig. 1, the two-phase training strategy of *Play it Straight* begins with a *boot* (warm-up) phase, in which a randomly initialized DNN model ϕ is partially trained on the entire dataset using the Repeated Random Sampling algorithm *RS2* defined in Okanovic et al. (2024), with a low reduction factor (high cost reduction). As confirmed by our experimental analysis, this boot phase can yield a sufficiently informed initialization of the model parameters, which allows us to provide the data instances with importance scores for guiding the subsequent data selection step.

The second, *fine-tune*, phase consists in repeatedly selecting small-sized instance subsets, based on their associated importance scores, and exploiting them to incrementally fine-tune the DNN model. This fine-tune loop ends when either the target accuracy, computed on a validation set (gathering labeled data instances, kept apart from the training ones), or the maximum energy budget stated by the user have been overcome.

Within this AL-inspired training process, a critical design choice lies in the strategy adopted for data selection at each fine-tuning round. Since the selection procedure may need to be applied to a large pool of candidate instances, it should introduce only minimal computational overhead while still ensuring the identification of highly informative examples.

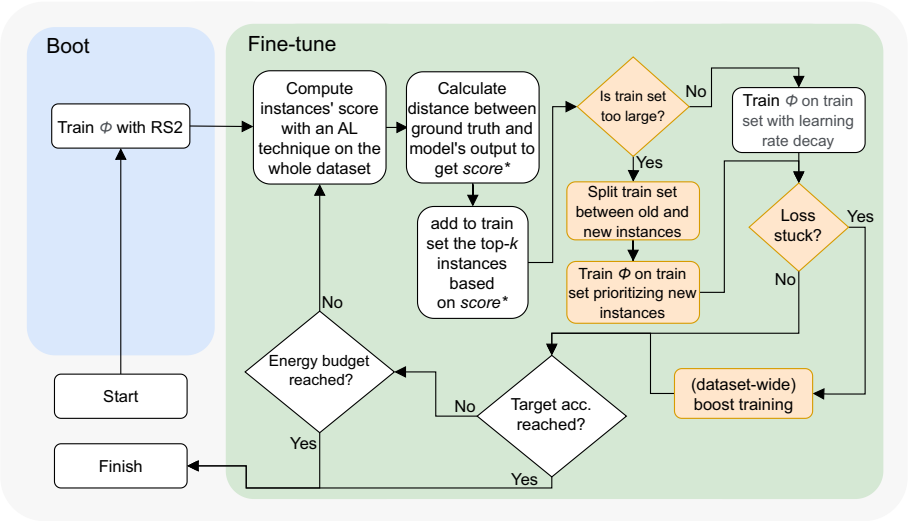


Fig. 1 Overview of the proposed framework: processing flow of algorithms *Play it Straight* and *Re-Play it Straight*. Specific additional components of *Re-Play it Straight* are highlighted in orange. Symbol ϕ here denotes the DNN model being trained

To this end, each data instance is assigned a combination of an uncertainty-based (namely, least confidence, entropy or margin) score with an error-based score, both of which are computationally inexpensive yet effective in guiding the selection process.

Algorithm *Re-Play it Straight* extends this basic two-phase computation by introducing several additional mechanisms into the fine-tune loop, for the sake of effectiveness and efficiency: (i) the training data volume at each cycle is kept bounded by using an extension of *RS2* over-sampling the instances that have been selected at that cycle by the data acquisition procedure; (ii) one additional "boost" training epoch is performed over the entire dataset whenever the training process seems to have converged to a (possibly non optimal or misled by serious distribution shift phenomena) stationary point of the loss function; (iii) in each training cycle, a cosine annealing schedule, with fixed initial and final values, is applied to the learning rate hyperparameter (controlling the size of gradient-driven parameter updates).

1.2.2 Novelty and significance of the proposal

The proposed framework advances the state of the art in the field along three main directions, which address the open issues and research gap discussed before:

- First, by coupling AL with *RS2*, it overcomes a key limitation of pruning and sampling approaches, namely the need to determine a-priori the number of epochs or data instance required to reach a target accuracy. This way, the iterative refinement enabled by AL allows the process to adapt dynamically, avoiding wasteful retraining.
- Second, the integration with *RS2*-like dataset-wide training steps (in both the boot phase and occasional boost epochs) makes the use of AL itself more energy-efficient. In particular, *RS2* mitigates the above-mentioned distribution shift issue (Yang et al., 2025; Okanovic et al., 2024).

- Third, the learning rate decay of *Re-Play it Straight* in each AL cycle results in a cyclic modulation of the learning rate across the fine-tuning phase, in the spirit of Smith (2017); Loshchilov and Hutter (2017); Goujaud et al. (2022)), while possibly performing additional (boost) training epochs over the dataset as a whole.

This original mix of algorithmic solutions is ultimately meant to improve time-to-accuracy (and energy saving) performances, by specifically leveraging AL data acquisition mechanisms and cyclical learning rates to accelerate the model learning progress (and possibly escape sub-optimal local minima) and additional dataset-wide (boost) training steps to help curb gradient estimation bias and stabilize the training dynamics.

Experiments performed on two benchmark datasets have shown that algorithms *Play it Straight* and, in a more pronounced way, *Re-Play it Straight* manage to substantially reduce the compute and energy consumed of state-of-the-art solutions in the field, in training a relatively large DNN model, without compromising accuracy performances. By empirically providing a positive answer to research question RQ1, our study expands the state of the knowledge in the field of data efficient DNN training, where several studies (Okanovic et al., 2024; Ayed & Hayou, 2023; Guo et al., 2022) recently supported the superiority of random data sampling with respect to importance-biased ones (like those used in most score-based coreset selection methods and in pure Active Learning methods). From a practical point of view, these results make us confident in the potentiality of the proposed framework to foster a more sustainable approach to the training of DNN models, which looks particularly important in Green-AI applications.

An abridged version of the framework proposed here was presented in Scala et al. (2025), where a preliminary version of *Play it Straight* was introduced. Compared to this earlier work, this manuscript makes three novel significant improvements: the extended version of *Play it Straight* (named *Re-Play it Straight*) proposed in the current submission introduces an adaptive mix of mechanisms that allow it to improve the performance and energy efficiency of *Play it Straight*; the evaluation test bed described in this manuscript includes a wider range of datasets and tasks than the one discussed in Scala et al. (2025); the use of different performance metrics makes the comparative analysis between these algorithms, and with the state-of-the-art *RS2*method, richer and more significant statistically, compared to the one presented in Scala et al.(2025).

1.3 Organization

The rest of this paper is structured as follows: After providing an overview of existing approaches in Sect. 2, the proposed methods *Play it Straight* and its extended version are illustrated in detail, in Sect. 3. The experimental study conducted to evaluate this method, comparatively with previous ones in the field, is then illustrated in Sect. 4. The concluding section finally discusses the main experimental findings and the main contributions of our research work, as well as some limitations of it and future research directions.

2 Related work

Our proposed framework, which integrates Active Learning with the *RS2* algorithm, aligns with the objectives of Green AI and introduces a novel approach to improving training efficiency for DNNs. This methodology builds upon a foundation of prior work across several related domains, each contributing to the current understanding of efficient model training. The following sections review key advancements in three areas: dataset pruning and sampling for efficient training, active learning-based training optimization and the usage of cyclic learning rates for improving DNN training. This contextualization positions our work within the broader research landscape and highlights its distinctive contributions. At the end of the section we summarize the novelties of the proposed approach w.r.t. previous works in these areas.

2.1 Dataset pruning/sampling for efficient model training

The significance of energy conservation has long been recognized (Garcia-Martin et al., 2019; Patterson et al., 2021; Xu et al., 2021), leading to ongoing advancements in power consumption estimation methodologies. Alongside these theoretical developments, practical tools for building energy consumption modeling have emerged. Traditional training methods use large data volumes in a monolithic training phase, which can be computationally expensive, especially for large datasets and complex models. In general, reducing the size of training data can then be a lever for cutting computational costs and energy consumption, at least linearly in the amount of pruned data. Indeed, dataset reduction methods (Yu et al., 2024; Zhao et al., 2021; Loo et al., 2022; Wang et al., 2022) have gained significant attention in machine learning due to their potential to improve model generalization and reduce computational costs. These methods can be divided into two main categories: *selection-based* and *synthesis-based* methods. *Selection-based* (a.k.a. *dataset pruning*) methods extract a small sample (or *coreset*) of the most relevant samples from the original dataset. *Synthesis-based* (a.k.a. *dataset distillation*) methods, on the other hand, create a new, smaller dataset by condensing the information from the original dataset. The synthesized data aim to accurately represent the original data distribution, even though they are not taken directly from the original data. For example, a class containing hundreds of images could be condensed into a single, more abstract yet information-rich image.

Although our proposal does not merely select a subset of data for model training, but rather employs a cyclic and adaptive selection strategy, it nonetheless shares certain similarities with DP techniques-particularly with *Play it Straight*. In contrast, *Re-Play it Straight* diverges more substantially from the conventional notion of DP.

Quite a wide variety of DP methods have been proposed in the literature (Guo et al., 2022), featuring different strengths and weaknesses. Over the years, numerous DP methods have been proposed. The most representative approaches can be broadly categorized into the following families:

- *Geometry Based Methods*: These approaches rely on the assumption that data points located in close proximity within the feature space tend to share similar properties. Consequently, geometry-based methods aim to eliminate data points that contribute redundant information, thereby constructing a coreset from the remaining, more informative

samples. Among these methods there is K-Center Greedy (Sener & Savarese, 2017);

- *Uncertainty Based Methods*: These methods are grounded in the intuition that samples on which the model exhibits lower confidence exert a greater influence on optimization compared to highly confident samples, and should therefore be prioritized for inclusion in the coreset (Coleman et al., 2020);
- *Error/Loss Based Methods*: Within a dataset, training samples are considered more critical if they contribute substantially to the model's error or loss during neural network training. The importance of a sample can be quantified through its loss value, its gradient magnitude, or its influence on the predictions of other samples throughout the training process. Samples with the highest estimated importance are then selected to form the coreset. Among these methods there are Gradient Normed (GraNd) and the Error L2-Norm (EL2N) (Paul et al., 2021);
- *Gradient Matching Based Methods*: Since deep neural networks are commonly optimized via (stochastic) gradient descent, a natural objective is to approximate the gradients computed over the full training dataset D using those derived from a carefully selected (weighted) subset S. The goal is to ensure that the gradients produced by S closely match those of D, thereby minimizing the discrepancy between the two. Among these methods there are methods like CRAIG (Mirzasoleiman et al., 2020);
- *Bilevel Optimization Based Methods*: Coreset selection can be naturally formulated as a bilevel optimization problem. In this framework, existing studies typically treat the selection of the subset (i.e. optimizing the samples S or the selection weights W) as the outer objective, while the optimization of the model parameters θ on S constitutes the inner objective. Representative approaches include GLISTER (Yang et al., 2024);
- *Submodularity Based Methods*: Submodular functions provide a natural way to quantify diversity and informativeness, making them a powerful tool for coreset selection through maximization. Numerous functions satisfy this property including GraphCut (Iyer et al., 2021), Facility Location (FL) and Log Determinant (Iyer et al., 2021);
- *Proxy Based Methods*: Many coreset selection approaches require training models on the entire dataset-sometimes multiple times-to compute features or evaluate certain metrics, which can be computationally expensive. To alleviate this cost, *Selection via Proxy* methods (Coleman et al., 2020; Sachdeva et al., 2021) have been proposed. These methods construct lighter or shallower versions of the target models as proxies, typically by reducing the number of hidden layers, narrowing feature dimensions, or limiting the number of training epochs. Coresets are then selected efficiently based on these proxy models, significantly reducing computational overhead.

We point out that many of these methods entail heavy computation in order to eventually identify a data sample that is both small and representative/useful enough for subsequently training a DNN model. Even though this cost could be amortized across several model training sessions (possibly including hyperparameter optimization), one must take full account for the energy impact of it.

Notably, recent empirical studies (Okanovic et al., 2024; Guo et al., 2022; Ayed & Hayou, 2023) revealed that a pure (uniform) random sampling scheme often allows for learning DNN models enjoying prediction performances quite similar (or even superior) to sophisticated Data Pruning/Selection methods, especially in the high-compression regime (Ayed & Hayou, 2023). Starting from this observation, recent efforts have been made to integrate

pure random sampling in training a neural-network model, to achieve a better trade-off between the representativeness and diversity of the sampled data and the efficiency of the training process as a whole. Such a strategy is at the core of the *Repeated Sampling of Random Subsets* (*RS2*) method proposed in Okanovic et al. (2024), which essentially consists in randomly sampling a subset of training data for each epoch. This method takes as input: a model to train, a dataset, the amount of epochs e, and a reduction factor r. Using this factor, it effectively performs $\frac{e}{r}$ training epochs, but basically executes e epochs with less data, leveraging a one-cycle cosine annealing learning rate scheduler. In general, however, extant Data Pruning and Sampling methods (including *RS2*) suffer from a drawback that may limit their practical value in some real-life applications: the user is required to carefully set a data-selection budget (i.e. a sampling percentage) beforehand in order to eventually achieve a desired level of accuracy. Hence, if this budget is set inadequately, it may be necessary to repeat the data-pruning and training processes as a whole, significantly increasing consumption.

Clearly, data pruning is not the only way to reduce model training costs. Other approaches to this task include, for example, modifying the sampling distribution during training, like in Katharopoulos and Fleuret (2018), where an importance sampling-based algorithm is introduced that accelerates training by exploiting a gradient-norm upper bound. An alternative approach consists ion scaling sample losses during training, like in Quercia et al. (2023), where the SGD optimization scheme is biased towards more important samples, identified after a few training epochs, by sampling them more frequently during the remaining training. In a similar vein, Chang et al. (2017) proposed to leverage lightweight estimations of sample uncertainty within SGD: variance in predicted probabilities and proximity to decision thresholds.

2.2 Efficient training via active learning (AL)

To the best of our knowledge, Salehi and Schmeink (2023) has been the first study on the application of Active Learning (AL) techniques to Green AI contexts, as a possible way to reduce the energy footprint of model training. Park et al. (2022) proposed to use an AL framework for data pruning, demonstrating its effectiveness but at a high energy cost without optimizations. In this context, some data acquisition methods widely used in AL studies, such as uncertainty sampling strategies (Settles, 2009), offer relatively low energy consumption. These strategies involve the model selecting data points it is most uncertain about for labeling, with the idea that labeling the most uncertain points provides the most information, improving performance with fewer labeled samples. Common uncertainty sampling criteria include Entropy, Margin and Least Confidence sampling (Settles, 2009). In fact, definitely higher computational burdens would be introduced by more sophisticated approaches, like the BAIT method proposed in Ash et al. (2021), which tries to optimize a bound on the Maximum Likelihood Estimators (MLE) error using Fisher information to guide batch sample selection. However, using a classical AL-based scheme as done in Salehi and Schmeink (2023) is unsuitable for Green-AI settings, owing to the high computational cost of repeatedly re-training (possibly up to convergence) a large DNN over increasingly larger data samples. Moreover, distribution shifts in the training data produced by AL cycles may lead to slow learning/convergence speed, so that more optimization steps will be needed to achieve the given accuracy target compared to a repeated random sampling approach like

that proposed in Okanovic et al. (2024) –and even compared to training on the full dataset in certain cases, as shown in Section 4.2.1.

The convergence of AL processes based on classical (non mini-batch) SGD and loss and uncertainty-based instance scores were recently studied in Haimovich et al. (2024), where a novel AL algorithm was proposed that is guaranteed to converge linearly in the number of acquired training instances. However, the convergence properties demonstrated in Haimovich et al. (2024) only hold under certain assumptions (including, primarily, the smooth convexity of the loss function), without ensuring fast convergence to good solutions in the general case. Moreover, the algorithm proposed in Haimovich et al. (2024) hardly fits real-life energy-efficient training applications where mini-batch SGD looks a more natural and more efficient solution for fully exploiting the parallel tensorial computation capability of contemporary GPUs/TPUs.

2.3 Cyclic learning rates for improving DNN training

As observed in Bartoldson et al. (2023), cyclic learning rate schedules have been shown both empirically and theoretically to produce speedups in time-to-accuracy performances, at least in certain scenarios (Loshchilov & Hutter, 2017; Goujaud et al., 2022). These schedules operate by systematically varying the learning rate in a periodic pattern throughout the training process, alternately raising and lowering the rate at regular intervals. The concept was initially introduced by Smith (2017), where several sawtooth-pattern scheduling approaches were introduced. Subsequently, Loshchilov and Hutter (2017) demonstrated that implementing cycles consisting of cosine decay patterns followed by restarts achieved competitive performance on image classification tasks. On the other hand, cyclic learning rate schedules have proven beneficial in escaping suboptimal stationary points, in the complex non-convex landscape of deep model losses, and in finding superior solutions, as well as in the discovery of temporal/snapshot ensembles, combining the output (Huang et al., 2017) or the weights (Izmailov et al., 2018) of models obtained at the end of different cycles. To the best of our knowledge, despite these nice properties of cyclic learning rates, no previous work in the field of data pruning and data-efficient DNN training has explored their combination with AL methods or with importance-based data selection/sampling —as proposed in our current work.

2.4 A brief comparison with the proposed approach

The approach proposed in this work differs from existing solutions in the field in several methodological and technical aspects.

Specifically, the core idea of training a model through multiple AL rounds (using at each round a small subset of data instances chosen based on cheap-to-compute loss/uncertainty scores) make our approach different from both Data Pruning and Coreset extraction methods (see Sect. 2.1 and from the algorithm *RS2* proposed in Okanovic et al. (2024). As a matter of fact, if at first sight this iterative shares some similarity with the repeated sampling

loop of *RS2*, our approach features several major distinguishing points: (i) In each AL round (consisting of multiple training epochs), we train the model on a biased data sample that includes the data instances for which it has made the highest prediction uncertainty/error (among those that have not been selected yet in previous AL rounds); by contrast, algorithm *RS2* keeps on training the model on uniformly sampled instance batches. (ii) In each AL round, the learning rate is made decay according to a cosine annealing pattern so that a cyclic learning rate schedule appears as different AL rounds unfold one after the other; in algorithm *RS2*, instead, the learning rate varies according to a cosine-annealing cycle only that spans from the first optimization step to the final one —notably, *RS2* needs to be provided with the total number of training epochs to perform, differently from our algorithms.

On the other hand, differently from the few AL-based approaches to efficient DNN training appeared in the literature (Salehi & Schmeink, 2023; Park et al., 2022), in our proposal several technical solutions are adopted to accelerate and stabilize the training dynamics and possibly explore different local optima: (i) In addition to training the model through multiple AL rounds, it is also made undergo dataset-wide *RS2*-like training epochs (in the boot phase and, in the case of algorithm *Re-Play it Straight*, in subsequent"boost" epochs). (ii) Algorithm *Re-Play it Straight* also enforces a bound on the number of training instances per AL bound, by introducing a "biased" variant of *RS2* where the newly selected instances are used more times than those that were used already in previous AL rounds.

In conclusion, a major trait of novelty of the proposed approach resides in the original mix of strategies and algorithmic solutions (repeated AL-like training on greedly-selected data, cyclical learning rate schedule, dataset-wide training epochs), which aim at synergistically minimizing the cost required to reach a desired level of model accuracy –while trying to curb the risk of slow convergence and overfitting that is known to possibly affect biased training data sampling/pruning strategies (Okanovic et al., 2024; Ayed & Hayou, 2023; Yang et al., 2025).

3 Proposed approach

Let $D \triangleq \{(x_i, y_i)\}_{i=1}^{n}$ be a given set of example data, such that each $x_i \in X$ is a data instance and each $y_i \in Y$ is the associated class label, represented as a one-hot vector in $[0,1]^C$ (i.e. a vector containing only one non zero element equal to 1, indicating the class label), where $C \in \mathbb{N}$ denotes the number of classes.

Let $\phi_\theta : X \to Y$ be a Deep Neural Network (DNN) classification model, parameterized by θ, that needs to trained on some subset D_s of D, and $l : Y^2 \times \theta \to \mathbb{R}^+$ be the loss function (continuous and twice-differentiable in θ) to be used to train ϕ_θ.

Notably, since $D \supset D_s$, training model ϕ_θ on D_s approximately enjoys a $\frac{|D|}{|D_s|}\times$ speedup per epoch, compared to training ϕ_θ on D_s, so leading to a substantial reduction of the total energy consumed in the training process when fixing the total number of epochs –even though there is no guarantee that such an efficiency improvement also holds for the total

time to convergence/accuracy, which depends on several other factors like the dataset distribution, mini-batch sizes and distributions, and optimization dynamics.

3.1 Algorithm play it straight

Algorithm 1 outlines our proposed approach, dubbed *Play it Straight*. This name reflects our hybrid strategy: we efficiently train a DNN model "*play it*" over subsets of informative labeled instances extracted from a given large dataset Don the basis of "*straight*" (i.e. easy-to-calculate with minor computation overhead) importance scores. In addition to the dataset D, *Play it Straight* takes the following arguments: a neural network model ϕ; the maximal numbers *bootEpcs* and *ftEpcs* of training epochs for the "boot" and "fine-tune" phase, respectively; a maximum energy budget *B*; a dissimilarity measure d over *Y*; the number k of instances to select at each fine-tune round; and an instance ranking function f_{rank}.

Algorithm 1 *Play it straight*

Data: D: dataset; V validation set; ϕ_θ: a neural network model (with randomly initialized parameters θ); *bootEpcs*: number of boot epochs; r: reduction factor for *RS2*; *ftEpcs*: number of fine-tune epochs; k: number of instances to select at each fine-tune round; f_{rank}: AL-like instance ranking function; d: a dissimilarity measure over Y; a_t: target accuracy; B: energy budget

$\phi_\theta \leftarrow$ *RS2*(ϕ_θ,D, *bootEpcs*, r) // Training the model ϕ_θ on D for *bootEpcs* epochs using algorithm *RS2* with reduction factor r)
$D_s \leftarrow \emptyset$
$D_u \leftarrow D$
$B_c \leftarrow 0$ *// variable storing the energy consumed by the algorithm*
while the validation accuracy of ϕ_θ on V is lower than a_t and energy $B_c < B$ **do**
  $S \leftarrow []$
  for $(x, y) \in D_u$ **do**
    $score^* \leftarrow f_{rank}\ (\phi_\theta(x))+d(y, \phi_\theta(x))$
    $S \leftarrow S \cup score^*$
  top-k$\leftarrow$ Select top-k instances from D_u based on the scores in S
  $D_u \leftarrow D_u \setminus$ *top-k*
  $D_s \leftarrow D_s \cup$ *top-k*
  Update the parameters θ of ϕ_θ via gradient descent over D_s for *ftEpcs* epochs
  Update B_c by adding, its current value, the energy consumed in this loop iteration
return ϕ_θ

The algorithm consists of two phases in a sequence:

1. *Boot phase*: First *Play it Straight* exploits the fast convergence of the *RS2* algorithm (Okanovic et al., 2024) to find an accurate enough preliminary setting for the parameters of ϕ(as shown in Okanovic et al. (2024), using *RS2* algorithm to this end is more effective than performing the same number of optimization steps using a traditional SGD-like procedure). However, as shown in our experimental study, *RS2* tends to experience a deceleration in accuracy gains. Thus, *Play it Straight* continues training ϕaccording to an AL-like procedure, in order to revitalize the training process and eventually achieve better model performance with acceptable energy consumption;
2. *Fine-tune loop*: Then, model $\phi$is made undergo an iterative fine-tune procedure, in each *round* of which k additional instances are selected from D, based on their importance scores (as explained in the following) and incorporated into D_s. During each fine-tune round, the model is updated and trained using both the newly-added instances and previously chosen ones. Throughout these rounds, an AL-oriented importance score is assigned to each instance x in $D_u \triangleq D \setminus D_s$ by using the chosen ranking function f_{rank}, combined with a dissimilarity measurement computed by applying the given function d to the model's output and the one-hot vector representing the ground-truth class of x. This allows us to assign an *enhanced importance score* ($score^*$) to x, which accounts for both the uncertainty and error associated with the prediction returned for x by the current version of the model being trained. The *top-k* instances from D_u are then selected based on these enhanced scores, added to D_s, and removed from D_u. Finally, the model is trained for *ftEpcs* epochs on the updated D_s. By adopting such an iterative and adaptive process, in place of a one-shot data pruning approach, we can significantly decrease the computation and energy costs, especially in the initial rounds where quite few data instances are used for model training. On the other hand, since the iterative refinement procedure proposed here can be stopped as soon as the desired accuracy level is achieved, our approach looks more flexible than typical data pruning (and core-set selection) methods, which require the user to "guess the right data reduction level" allowing to achieve the desired accuracy while minimizing the computation cost.

Once completed the loop, the fully-trained version of model ϕ_θ is returned.

Please notice that as soon as the energy budget is depleted, the algorithm stops and returns a model with the current parameter configuration (i.e. the one computed in the last completed iteration of the main loop in the algorithm). In such a case, there is no guarantee on how close the actual accuracy of the returned model is to the target accuracy. In other words, considering the energy bound as a strict constraint, the algorithm just returns a sort of "best-effort" solution, which it managed to find without infringing that constraint.

3.2 Algorithm re-play it straight

Data: D: dataset; V validation set; ϕ_θ: a neural network model (with randomly initialized parameters θ); *bootEpcs*: number of boot epochs; r: reduction factor for *RS2*; d_b: discount factor for the boot phase; *ftEpcs*: number of fine-tune epochs; k: number of instances to select at each fine-tune round; f_{rank}: AL-like instance ranking function; d: a dissimilarity measure over Y; a_t: target accuracy; B: energy budget

if $d_b \neq 0$ **then**
  $D_{boot} \leftarrow randomSample(D, d_b)$ randomly select $1/d_b$ of the original dataset
else
  $D_{boot} \leftarrow D$
$\phi_\theta \leftarrow RS2\ (\phi_\theta, D_{boot}, bootEpcs, r)$ // Train ϕ_θ on D for *bootEpcs* epochs using algorithm *RS2* with reduction factor r
$D_s \leftarrow \emptyset$, $D_u \leftarrow D$, $B_c \leftarrow 0$
while the validation accuracy of ϕ_θ on V is lower than a_t and energy $B_c < B$ **do**
  $S \leftarrow []$, reset the learning rate to the initial (top) value
  for $(x, y) \in D_u$ **do**
    $score^* \leftarrow f_{rank}\ (\phi_\theta(x)) + d(y, \phi_\theta(x))$
    $S \leftarrow S \cup score^*$
  top-k$\leftarrow$ Select top-k instances from D_u based on the scores in S
  $D_u \leftarrow D_u \setminus$ *top-k*
  if $\mid D_s \mid \geq$ *25,000* **then**
    $Splits_{D_s} \leftarrow split\left(D_s, min\left(\frac{|D_s|}{k}, ftEpcs\right)\right)$
    for e *in ftEpcs* **do**
      $j \leftarrow e\ \%\ |Splits_{D_s}|$ // % stands for standard modulus operator
      $D_{current} \leftarrow$ *top-k* $\cup\ D_{split}[j]$
      Update θ of ϕ_θ via gradient descent[1] over $D_{current}$ for 1 epoch keeping on decreasing the learning rate with cosine annealing. Let L_{pre} and L_{cur} the loss values before ad after this step, respectively.
      if *stuckedLoss*(L_{pre}, L_{cur}) **then**
        $\phi_\theta \leftarrow RS2\ (\phi_\theta, D, 1, 1)$ % run 1 epoch of training on D
    $D_s \leftarrow D_s \cup$ *top-k*
  else
    $D_s \leftarrow D_s \cup$ *top-k*
    Update the parameters θ of ϕ_θ via gradient descent[1] over D_s for *ftEpcs* epochs using cosine-annealing learning rate decay
  Update B_c by adding, its current value, the energy consumed in this loop iteration
return ϕ_θ

Algorithm 2 *Re-play it straight***Algorithm 3** split

Data: D_s: dataset to split; n: amount of splits
if $|D_s| = 0$ **then**
  return [ ]
splits $\leftarrow$ [ ]
$size_{splits} \leftarrow \left\lfloor \frac{|D_s|}{n} \right\rfloor$
$D_s \leftarrow randomShuffle(D_s)$
for $i \leftarrow 0$ **to** $n - 1$ **do**
  $D_{subset} \leftarrow D_s[i \cdot size_{splits} : (i + 1) \cdot size_{splits}]$
  append D_{subset} to *splits*
return *splits*

We enhanced the base algorithm introduced in Sect. 3.1 to address some of its limitations. Specifically, we applied the following updates to *Play it Straight* in order to improve its performances:

1. Reduction of Training Data per Round: Algorithm 2 lines 14 – 21. A significant drawback of *Play it Straight* is that as it progresses through AL-like rounds, the volume of training data (quadratic in the number of rounds) can grow excessively, leading to high computational costs. To mitigate this, we control the size of training data in every round by fixing a maximal threshold for how many of the instances selected in previous rounds can be kept. This threshold was empirically set to 25,000 samples, as we observed that this choice allows for striking a satisfactory balance between efficiency and model generalization[1][2]. We devised a modified version of algorithm *RS2* to suitably mix the "old" data instances coming from previous rounds with the new ones added in the current round, so that the model's "memory" is effectively "refreshed" with old data while consistently incorporating new samples. After splitting the old data instances into chunks, through function *split* (line 15 of Algorithm 2), in each epoch of the current round the model is fine-tuned using a distinguished chunk of the old data along with all the new samples. More specifically, while using all the fresh instances selected in the current AL round, in each epoch only a portion of the old data instances (selected in previous rounds) is retained according the following scheme: (*i*) if the old instances are less than the product between the number of new data instances and the number of epochs, they are divided into chunks of size k (i.e., the same size as the newly selected data sample); (*ii*) otherwise, the old data instances are divided into as many data chunks as the number of epochs in the current round, to ensure that each old data instance is seen again at least once. Algorithm 3 presents the split function, devoted to split a dataset D_s into n chunks. It first computes the size of each split as $\lfloor \frac{|D_s|}{n} \rfloor$, then randomly shuffles the old data to prevent samples from appearing in the same order, and finally divides them into splits. This approach reduces the frequency of backward operations by limiting the model's exposure to old data, while still ensuring it does not forget them, as they are periodically reintroduced;

2. Boost Epochs: Algorithm 2 lines 20 – 21. To address the issue of slow convergence during training, especially when using small datasets, we introduced boost epochs. As a matter of fact, we observed that the reduction of the amount of training data, described above when introducing the *Reduction of Training Data per Round*, may result in a slow down of the training algorithm convergence rate, thereby increasing training costs. Hence, we introduced a new phase which checks whether the loss between training epochs within AL-like rounds is not significantly reduced. If this happens, the algorithm performs an additional training epoch using the *RS2* algorithm on the whole dataset – the *boost* epoch– before resuming the standard procedure. These epochs use a learning rate set to 10% of the current rate in the AL-like round, allowing the model to refresh

[1] A variant of mini-batch Stochastic Gradient Descent (SGD) procedure is used here to perform gradient descent optimization. Details on this respect can be found in Section 4.

[2] In fact, we noted that *Play it Straight* looses its energy efficiency considerably when the number of samples exceeded this threshold. In the enhanced *Re-Play it Straight* this specific threshold value was empirically chosen as a way to ensure a satisfactory trade-off between two contrasting risks: on one hand, if the model is trained on too few samples (i.e., less than 25,000) it is likely to overfit the newly-selected data; on the other hand, when exceeding this threshold, the computational cost is likely to increase considerably with minor performance improvements.

its memory without deviating significantly. This approach is employed because when using *RS2*, a Cosine Annealing learning rate is used, which starts with a high value and gradually decreases towards the end of the training. While this high initial learning rate can help the model learn better during the early stages of training, it can also cause significant deviations in the model's behavior during later phases. By lowering the learning rate in this phase, we prevent the model from losing the acquired knowledge while still allowing it to refine and adapt its understanding without overfitting, while also enabling the model to deviate from local minima and explore potentially better solutions. This helps guiding the model toward a minimum more efficiently. Function *stuckedLoss* (Algorithm 2 line 20) checks the whether the loss is not significantly reduced. To perform this check, any criterion for empirically assessing the convergence of iterative deep learning procedures can be adopted. In the current implementation of algorithm *Re-Play it Straight*, this is done by simply checking whether the difference of the losses between the epochs is below a threshold;

3. Lightened Boot Phase: Algorithm 2 line 5. We introduced a new parameter called the "boot discount factor", which reduces the amount of data used during the boot phase and boost epochs by randomly selecting a subset of the data. This can be viewed as an enhancement or variation of the *RS2* algorithm, as the random selection is applied each time the *RS2* algorithm reshuffles the data. In many cases, using fewer data points during these phases does not lead to performance degradation, making this parameter an effective way to improve computational efficiency;
4. Cyclic learning rate: In *Re-Play it Straight*, the scheduling of the learning rate (in gradient-descent optimization steps) is a critical component, as multiple mechanisms interact throughout the training process. During the boot phase, the learning rate is initialized and updated following the *RS2* scheme. Once the cyclic phase begins, the scheduler performs a single cosine annealing cycle in each Active Learning (AL) round (the period of each learning rate cycle is $ftEpcs \times nbatch$, where *nbatch* is the number of batches composing the current training set $D_{current}$ or D_s). In boost epochs, the learning rate is kept constant; subsequently, it decays monotonically (according to a cosine-annealing law) from its maximum to minimum value, and is then reset. Learning rate adjustments are applied at every optimization step.

3.3 Setting guidelines and implementation choices

Active learning (AL) offers a pathway to streamline AI model development while aligning with the principles of Green-AI. The core concept lies in the strategic selection of the most informative data samples from a larger labeled dataset. In principle, by using only such a small subset of samples for model training, the total computational costs needed to reach a predefined target accuracy level can be reduced. However, the amount of energy saving that can be obtained strongly depends on the following factors:

- *Data reduction effectiveness*: A core measure of AL effectiveness is its ability to drastically reduce the training set size while preserving model performance. The greater the reduction achievable, the higher the potential energy savings;
- *Data sampling complexity*: Data sampling methods proposed in AL literature differ a lot in their computational overhead. Simpler methods like uncertainty sampling have

minimal cost, while more sophisticated approaches entail heavier compute. Indeed, using some computationally intensive AL technique may render the proposed method ineffective, because the selection process can become more burdensome than the neural network's training;
- *Impact on training convergence*: The interaction between data reduction and the model's convergence behavior cannot be ignored. In some cases, a highly informative dataset might lead to fewer training iterations, amplifying savings. However, it's also possible that more iterations might be required to converge, partially offsetting the energy gains.

In the current implementation of the approach, we have considered three alternative uncertainty-based criteria to instantiate the function f_{rank} attributing importance scores to the data instances, in order to incrementally select a subset of those achieving the highest scores:

- *Least confidence* (denoted hereinafter as *lc*, for short): Let p be the probability of the most likely class for a data instance x. Then the least confidence score assigned to x is simply computed as $1 - p$;
- *Margin sampling* (referred to as *margin* from now on): This criterion focuses on the difference between the probability of the most likely class and the second most likely class. If, for a data instance x, p_{top1} and p_{top2} are the probability of the most likely class and of the second most likely class, respectively, then the margin score of x is computed as $p_{top1} - p_{top2}$;
- *Entropy* (simply denoted as *entropy* hereinafter): Entropy measures the overall uncertainty across all classes. A high entropy value indicates the model is unsure about the correct class. For a data instance x, if there are C classes and p_i is the probability of the i-th class, the entropy is calculated as $-\sum_{i=1}^{C} p_i \log p_i$.

4 Experimental evaluation

4.1 Test bed

4.1.1 Datasets

We used the following datasets to execute the experimental evaluation:

- *CIFAR-10* (Krizhevsky et al., 2009): which consists of 60000 instances representing 32x32 colour images, labeled using 10 mutually exclusive classes, with 6000 images per class. The dataset is organized into 50000 instances as the training set and 10000 instances as the test set. The latter contains 1000 randomly-selected images from each class, while the training set is comprised of 5 training batches, each containing 5000 images per class;
- *CIFAR-100* (Krizhevsky et al., 2009): which consists of 60000 instances representing 32x32 colour images, labeled using 100 mutually exclusive classes, with 600 images per class. The dataset is organized into 50000 instances as the training set and 10000 instances as the test set. The latter contains 100 randomly-selected images from each

class, while the training set is comprised of 5 training batches, each containing 500 images per class. Additionally, we have created two variations of this dataset, one with 50 classes and another with 20 classes, maintaining the same structure but reducing the number of categories accordingly;

- *Imagewoof*: This is a subset of the ImageNet (Deng et al., 2009) dataset that contains images hard to be classified, as they're all dogs' images of various breeds. It consists of 12,954 color images at a resolution of 256x256 pixels. The dataset is divided into a training set of 9,025 images and a test set of 3,929 images. Imagewoof was specifically curated to test the robustness of image classification models, as distinguishing between similar dog breeds presents is more challenging than generic object classification tasks.

4.1.2 Terms of comparison and evaluation setting

We benchmarked *Play it Straight* against standard full-dataset training (referred to hereinafter as *Standard train*), the *RS2* algorithm proposed in Okanovic et al. (2024), the pure AL-based approach presented in Park et al. (2022) (we reported results only with the Margin measure in order to avoid overloading the tables and figures, as the other measures yield comparable outcomes) and state-of-the-art Data Pruning (DP) methods Glister, GraphCut, CRAIG, GraNd, by leveraging the respective implementations available in library DeepCore (Guo et al., 2022). In each test, we evaluated each of these methods by measuring both the total amount of energy (measured in Wh) it consumed and the accuracy of the models discovered. Inspired by the time-to-accuracy analysis conducted in Okanovic et al. (2024), we fixed different accuracy targets (namely, from 60% to 90% on CIFAR-10, and from 50% to 75% on CIFAR-100)[3] and measured the amount of energy consumed by each method to reach each target —unless the method had exhausted its budget of energy/epochs before reaching the target.

We evaluated *Re-Play it Straight* on all the previously described datasets, comparing its performance exclusively with *RS2* and *Play it Straight*. These two methods were chosen as benchmarks due to their superior performance relative to other techniques. To assess computational efficiency, we measured the number of backward operations performed during training, which constitutes the main computational burden of model training. This approach was selected because the evaluated techniques focus purely on optimizing the training process, unlike data pruning methods, which incur significant energy costs during the pruning phase before training begins. Additionally, forward operations used to compute AL-like metrics were included in the analysis but assigned a weight of 0.5 relative to backward operations. This weighting accounts for the fact that backward propagation involves both error propagation and gradient computation at each layer, making its computational cost approximately twice that of forward propagation. In fact, for most layers, the number of floating-point operations (FLOPs) required for backward propagation is double that of forward propagation.

All the experiments were run on an Intel Xeon CPU E5-2698 v4 @ 2.20GHz, 250GB RAM, with Tesla V100-DGXS-32GB GPU. Energy measurements were made by using library *CodeCarbon*, version 2.4.1 (Courty et al., 2024).

[3] These ranges were chosen differently, starting from the different accuracy scores that the *Standard train*, energy-unaware, baseline obtained on the two datasets.

Table 1 Energy consumption (Wh) required to achieve various target accuracies on the CIFAR-10 dataset using different techniques

Target	60%	65%	70%	75%	80%	85%	90%
Standard train	20	39	59	78	157	626	1018
AL (margin) Park et al., (2022))	135	218	288	357	711	1167	2421
GraNd Paul et al., (2021)	838	865	878	902	924	1042	1232
Craig Mirzasoleiman et al., (2020)	175	196	231	248	344	442	627
Glister (Yang et al., 2024)	108	117	133	172	192	380	599
GraphCut Iyer et al., (2021)	239	364	396	486	621	762	993
RS2 w/o repl 20%	32	39	48	65	82	168	237
RS2 w/o repl 10%	44	59	75	97	108	153	197
RS2 w/o repl 5%	43	56	62	77	94	121	–
Play it Straight (margin)	**19**	**38**	42	47	**63**	**94**	**196**
Play it Straight (entropy)	**19**	39	**41**	**42**	**63**	99	220
Play it Straight (lc)	**19**	**38**	**41**	**42**	**63**	98	204

Lower values indicate greater energy efficiency. The best method(s) is bolded

Table 2 Energy consumption (Wh) required to achieve various target accuracies on the CIFAR-100 dataset using different techniques

Target	50%	55%	60%	65%	70%	75%
Standard train	101	395	622	1375	1606	1837
AL (margin)	1057	1304	1870	3265	5000	–
RS2 w/o repl 20%	107	189	210	250	304	–
RS2 w/o repl 10%	112	124	146	167	–	–
RS2 w/o repl 5%	75	80	95	–	–	–
Play it Straight (margin)	**65**	72	**94**	**140**	**199**	418
Play it Straight (entropy)	**65**	72	105	150	212	527
Play it Straight (lc)	**65**	**70**	103	149	212	**362**

Lower values indicate greater energy efficiency. The best method(s) is bolded

4.1.3 Hyperparameter configuration

In each tests, a ResNet18 (He et al., 2016) classification model was trained by using mini-batch Nesterov's Accelerated Gradient Descent (Botev et al., 2017), with an initial learning rate of 0.1 and a momentum of 0.9, and Cross-Entropy loss.

In all the runs algorithms *Play it Straight* and *Re-Play it Straight*, we always instantiated the function hyperparameter d (for measuring the dissimilarity between class-membership probability distributions) with standard Euclidean distance.

We tested *Play it Straight* using three variants of the ranking function f_{rank} (cf. Algorithm 1), associating each data instance to its *margin*, *entropy* and *lc* (i.e. least confidence) scores, respectively (see Sect. 3.3 for a definition of these scores). However, it is important to note that our approach is flexible and can accommodate other AL techniques or combinations thereof. After testing various dissimilarity measures, we eventually decided to only show here the results obtained with the L2 distance, seeing as the other measures tested did not improve these results appreciably.

As to the specific configuration of algorithm *Play it Straight*, in the boot phase, the *RS2* procedure was always run with a data reduction factor (per epoch) of 30% (i.e. $r = 0.3$)

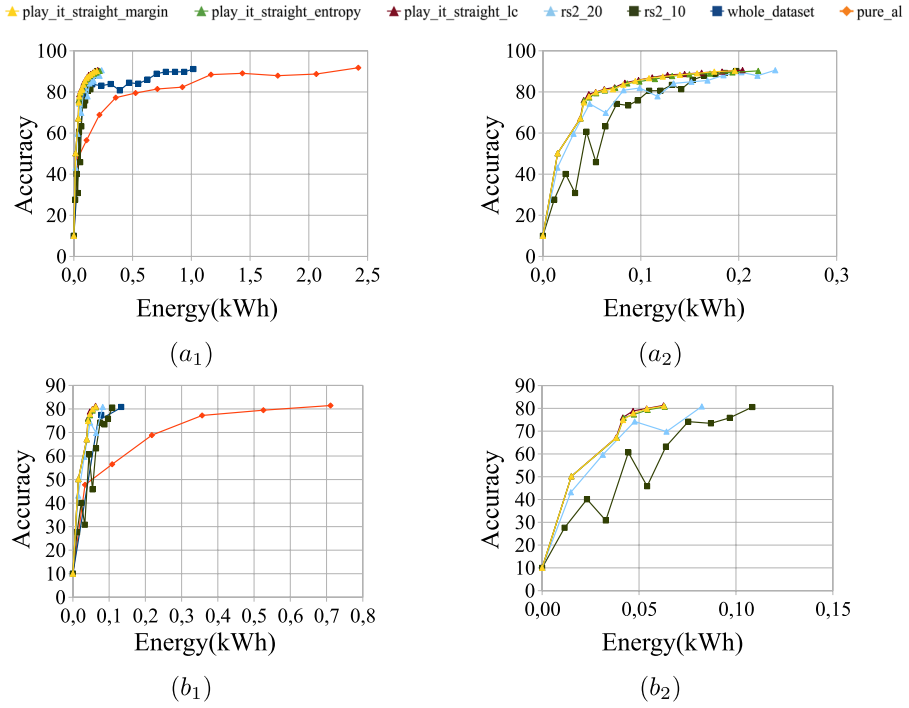


Fig. 2 Energy-to-accuracy for *Play it Straight* compared to *RS2*, AL, and standard training of ResNet18 on the full CIFAR-10 dataset, targeting 90% (**a**) and 80% (**b**) accuracy. Values are reported every 10 epochs. Subfigures (a_1, b_1) showcases all techniques, while (a_2, b_2) focuses on the low-energy methods

while fixing a maximum of 20 epochs (i.e. $\mathrm{bootEpcs} = 20$); in each fine-tune round, we made *Play it Straight* select 1000 instances for CIFAR-10 ($k = 1000$) and 5000 instances ($k = 5000$) for CIFAR-100, and run 10 optimization epochs ($\mathrm{ftEpcs} = 10$) for CIFAR-10 and 5 epochs ($\mathrm{ftEpcs} = 5$) for CIFAR-100. Regarding the specific configuration of *Re-Play it Straight*, the *RS2* procedure was always executed with the same settings as in *Play it Straight* during the boot phase. In each fine-tuning round, *Play it Straight* selected 1000 instances for CIFAR-10 ($k = 1000$), 7000 instances for CIFAR-100 ($k = 7000$), 4000 instances for CIFAR-50 ($k = 4000$), 3000 instances for CIFAR-20 ($k = 3000$), and 6000 instances for Imagewoof ($k = 6000$). The number of optimization epochs was set to 10 for CIFAR-10 ($\mathrm{ftEpcs} = 10$), 7 for CIFAR-100 ($\mathrm{ftEpcs} = 7$), 5 for CIFAR-50 ($\mathrm{ftEpcs} = 5$), 4 for CIFAR-20 ($\mathrm{ftEpcs} = 4$), and 3 for Imagewoof ($\mathrm{ftEpcs} = 3$). Additionally, the *RS2* discount factor was set to 2 for CIFAR-10 ($d_b = 2$), 1 for CIFAR-100 ($d_b = 1$), 1.5 for CIFAR-50 ($d_b = 1.5$), 1.5 for CIFAR-20 ($d_b = 1.5$), and 1 for Imagewoof ($d_b = 1$). We assumed an infinite energy budget B to evaluate the performance of our proposal without interruptions until completion. In a production environment, this can be adjusted based on specific requirements.

The hyperparameters of *RS2* (Okanovic et al., 2024) and the pure Active Learning (AL) method of Park et al. (2022) were favorably set following the papers in which they were proposed. Specifically, the AL method was configured to perform 20 AL rounds, and to select 1000 data instances per round based on Margin scores; at each of these AL rounds, the

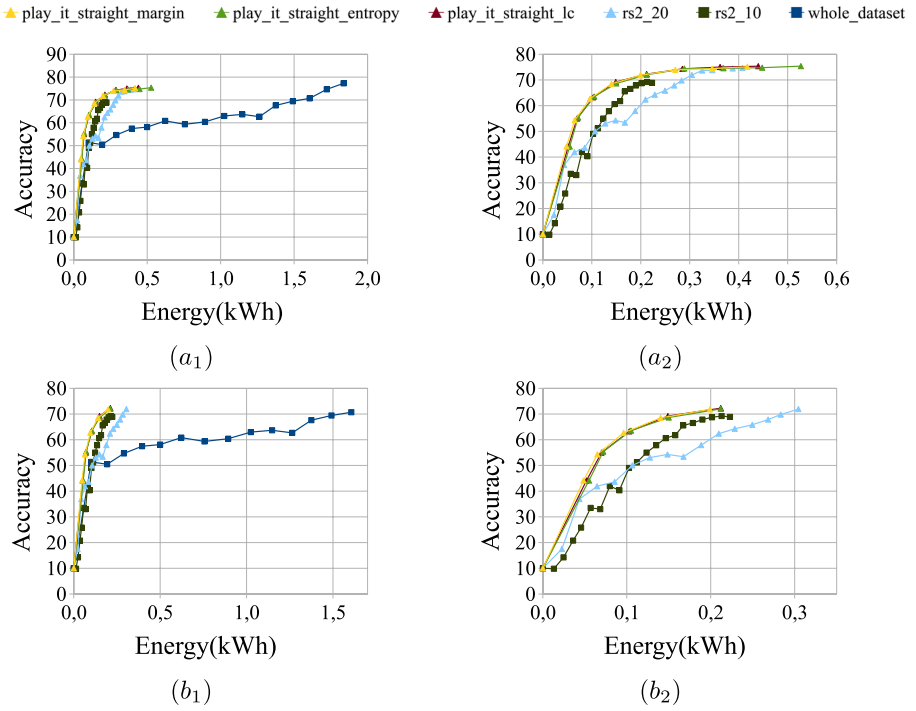


Fig. 3 Energy-to-accuracy for *Play it Straight* compared to *RS2*, AL, and standard training of ResNet18 on the full CIFAR-100 dataset, targeting 75% (**a**) and 70% (**b**) accuracy. Values are reported every 10 epochs. Subfigures (a_1, b_1) showcases all techniques, while (a_2, b_2) focuses on the low-energy methods

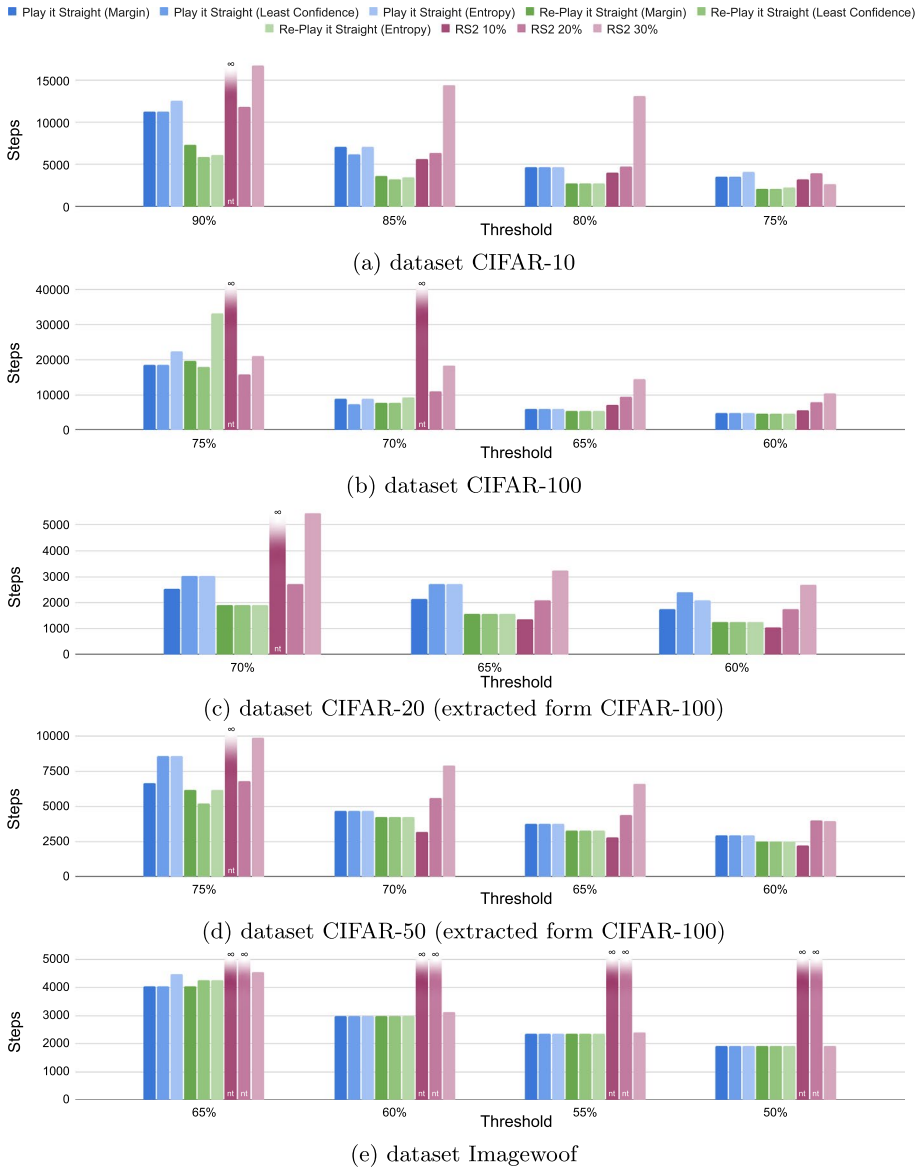
model was re-trained from scratch, for 200 epochs, over all the data instances accumulated up to that moment as done in (Park et al., 2022). Algorithm *RS2* was tested with different values of the reduction factor (namely 30%, 20%, 10% and 5%), considering a total budget of training epochs of 200, as proposed in Okanovic et al. (2024).

4.2 Test results

The following paragraphs present the experimental evaluation results for *Play it Straight*, highlighting its performance and key findings. Then is discussed the results for its extension, *Re-Play it Straight*, demonstrating its improved effectiveness. This comprehensive comparison provides insights into the strengths of both approaches and areas for potential improvement.

4.2.1 Play it straight

The analysis focuses on three key aspects: computational savings, accuracy, and pruning ratio, comparing the performance of *Play it Straight* with the previously explained techniques. A significant advantage of *Play it Straight* is its iterative approach to data selection: it eliminates the need to pre-determine the amount of data to prune. Instead, it dynamically adds only the data necessary to reach the target accuracy. The intelligent data selection of



(a) dataset CIFAR-10

(b) dataset CIFAR-100

(c) dataset CIFAR-20 (extracted form CIFAR-100)

(d) dataset CIFAR-50 (extracted form CIFAR-100)

(e) dataset Imagewoof

Fig. 4 Optimization steps needed to achieve different accuracy targets when using algorithms "*Play it Straight* " (blue bars) and "*Re-Play it Straight* " (green bars) with different data selection strategies (namely, Margin, Least Confidence, and Entropy), and algorithm *RS2* (magenta bars) with different settings of the reduction factor (namely, 10%, 20% and 30%). The shaded bars marked with **nt** in the chart indicate that the respective method could not reach the target accuracy

Play it Straight enables it to reach the target accuracy more quickly, translating to computational savings despite potentially higher *pruning ratio* (*pr*) compared to other methods.

As easily seen in Tables 1 and 2, *Play it Straight* outperforms the other techniques in terms of computational savings given the same target accuracy, demonstrating the effective-

ness of its iterative data selection strategy. This suggests that the “better" data selected by *Play it Straight* not only speeds the training process up but also allows it to achieve all the considered target-accuracy levels. Notably, the latter nice property is not enjoyed by the other methods analyzed (excluding the standard train baseline, performing no kind of data reduction), which fail to meet some of the target accuracy thresholds.

Figure 2 for CIFAR-10 and Fig. 3 for CIFAR-100 illustrate the relationship between energy consumption (x-axis) and accuracy (y-axis). These figures clearly demonstrate that our proposed technique can achieve the target accuracy with a reduced energy consumption compared to standard training, AL, *RS2* and so on. Furthermore, in Fig. 3, the line corresponding to the pure AL approach is not reported, since its convergence is very slow and its inclusion would have made the results of the other techniques unreadable. Nevertheless, the corresponding values are reported in Table 2 for completeness. Notably, as dataset complexity increases, the energy savings achieved by our method become more pronounced across both target accuracy levels.

We note that while precision, recall, and F1 score were computed for a comprehensive evaluation, they are not detailed here for lack of space, considering that their trends closely mirror those of accuracy. This let us focus on accuracy as a representative metric without undermining the value of this empirical study.

4.2.2 Re-play it straight

In the experimental evaluation, we compared the number of steps required to achieve some predefined accuracy thresholds. with the term we mean backward passes within the model, while forward only passes are taken into account as follows: we assume forward passes to be half the cost of backward passes, the reason is intuitive: in neural networks, most of the computational cost comes, for simplicity’s sake, from Linear or Convolution layers, which perform operations of the form:

$$y = W \cdot x \tag{1}$$

where W are the trainable parameters.

The forward pass requires one matrix multiply ($W \cdot x$), while the backward pass must compute gradients for both the weights (∇W) and the inputs (∇x), each requiring a matrix multiply. Therefore, the backward pass is approximately twice as costly as the forward pass.

As anticipated, *Re-Play it Straight* consistently outperforms the other methods in most scenarios. The results are closely tied to the specific AL-like technique employed, but based on the data, it is clear that the Margin-based approach performs better compared to the others.

We observed that the performance of the method degrades as the number of classes increases (see Fig. 4). This is primarily due to the fact that with a larger number of classes, the amount of representative data for each class per round becomes significantly limited. While increasing the amount of selected data could mitigate this performance gap, it would also negate the advantage of using less data for training. To demonstrate this, experiments were conducted by training the models in different settings with varying numbers of classes.

Overall, *Re-Play it Straight* performs on par with or better than the original version. This is the case, indeed, of *RS2*, which despite seeming to perform slightly better in some cases,

Table 3 This table presents the effectiveness values of the proposed techniques.

Dataset	Acc.	*Play it Straight*			*Re-Play it Straight*			*RS2*		
		Mar.	lc	Ent.	Mar.	lc	Ent.	10%	20%	30%
CIFAR10	90%	0.80	0.80	0.72	1.23	**1.53**	1.47	n/a	0.76	0.54
	85%	1.21	1.37	1.21	2.36	**2.63**	2.49	1.52	1.34	0.59
	80%	1.71	1.71	1.71	**2.91**	**2.91**	**2.91**	2.00	1.69	0.61
	75%	2.14	2.14	1.85	**3.56**	**3.56**	3.31	2.34	1.90	2.86
CIFAR100	75%	0.41	0.41	0.33	0.38	0.42	0.23	n/a	**0.47**	0.36
	70%	0.79	0.95	0.79	**0.90**	**0.90**	0.75	n/a	0.63	0.38
	65%	1.09	1.09	1.09	**1.19**	**1.19**	**1.19**	0.90	0.69	0.45
	60%	1.25	1.25	1.25	**1.28**	**1.28**	**1.28**	1.07	0.76	0.57
Imagewoof	65%	**0.29**	**0.29**	0.26	**0.29**	0.27	0.27	n/a	n/a	0.26
	60%	**0.36**	**0.36**	**0.36**	**0.36**	**0.36**	**0.36**	n/a	n/a	0.35
	55%	**0.42**	**0.42**	**0.42**	**0.42**	**0.42**	**0.42**	n/a	n/a	0.41
	50%	**0.47**	**0.47**	**0.47**	**0.47**	**0.47**	**0.47**	n/a	n/a	0.47
CIFAR50	75%	0.56	0.44	0.44	0.61	**0.72**	0.61	n/a	0.55	0.38
	70%	0.75	0.75	0.75	0.83	0.83	0.83	**1.09**	0.63	0.44
	65%	0.87	0.87	0.87	0.99	0.99	0.99	**1.16**	0.74	0.49
	60%	1.01	1.01	1.01	1.20	1.20	1.20	**1.36**	0.75	0.76
CIFAR20	70%	0.55	0.46	0.46	**0.74**	**0.74**	**0.74**	n/a	0.51	0.26
	65%	0.61	0.48	0.48	0.82	0.82	0.82	**0.96**	0.63	0.40
	60%	0.68	0.50	0.58	0.95	0.95	0.95	**1.15**	0.68	0.44

Acc. stands for Target Accuracy, Mar. for Margin, lc for Least Confidence, and Ent. for Entropy. Values are expressed in percentage. Bold values indicate the best values for combination of accuracy target and dataset

it fails to achieve high accuracy levels in those settings, unlike *Play it Straight* and *Re-Play it Straight*. The shaded bars in the Fig. 4 marked with **nt** in the chart indicate that the respective method could not reach the target accuracy. Additionally, the results on the challenging Imagewoof dataset, Fig. 4e, are particularly noteworthy. In this case, two settings of *RS2* even failed to reach the minimum accuracy threshold, while the most efficient setting that did achieve the threshold required a higher number of steps compared to *Play it Straight*. The boost provided by *Re-Play it Straight* is significant, but a more detailed discussion on this respect is provided in the ablation-study Sect. 4.3.

4.2.3 Comparison between play it straight and re-play it straight

As previously discussed, one key advantage of *Re-Play it Straight* over *Play it Straight* is its ability to intelligently reduce the amount of data used in AL rounds when the number of training samples becomes too high. As shown in Fig. 6, once the threshold of 25,000 samples is reached, the algorithm starts keeping fixed the total number of training instances that will be used in each of the subsequent fine-tune rounds. Notably, this comes with a neat efficiency improvement (in terms of required memory, compute and energy): while the total number of optimization operations performed by *Play it Straight* follows a quadratic trend, these (energy-dominant) operations are ensured to grow in a smoother, linear, way in the case of *Re-Play it Straight*, once the above-mentioned threshold has been reached. This allows, indeed, the latter algorithm to overcome a major limitation of the *Play it Straight* method originally proposed in Scala et al. (2025).

Table 4 Evaluation of *Re-Play it Straight* and *RS2* across multiple datasets of varying complexity

Dataset	Acc.	Algorithm	n. steps	Accuracy	Precision	Recall	F1
CIFAR10	90%	*Re-Play it Straight*	$7364_{\pm 815}$	$91\%_{\pm 0,7}$	$0,9_{\pm 0,006}$	$0,9_{\pm 0,007}$	$0,9_{\pm 0,007}$
			(11,1%)	(0,7%)	(0,7%)	(0,7%)	(0,7%)
		RS2	9400	$91\%_{\pm 0,4}$	$0,9_{\pm 0,004}$	$0,9_{\pm 0,004}$	$0,9_{\pm 0,004}$
				(0,4%)	(0,4%)	(0,4%)	(0,4%)
CIFAR100	75%	*Re-Play it Straight*	$5368_{\pm 686}$	$78\%_{\pm 4,2}$	$0,8_{\pm 0,040}$	$0,8_{\pm 0,042}$	$0,8_{\pm 0,042}$
			(12,8%)	(5,3%)	(5,1%)	(5,3%)	(5,4%)
		RS2	9400	$68\%_{\pm 0,9}$	$0,7_{\pm 0,009}$	$0,7_{\pm 0,009}$	$0,7_{\pm 0,009}$
				(1,3%)	(1,3%)	(1,3%)	(1,3%)
CIFAR20	75%	*Re-Play it Straight*	$1704_{\pm 283}$	$74\%_{\pm 1,6}$	$0,7_{\pm 0,017}$	$0,7_{\pm 0,016}$	$0,7_{\pm 0,017}$
			(16,6%)	(2,2%)	(2,3%)	(2,2%)	(2,3%)
		RS2	2000	$59\%_{\pm 1,9}$	$0,6_{\pm 0,020}$	$0,6_{\pm 0,019}$	$0,6_{\pm 0,019}$
				(3,3%)	(3,3%)	(3,3%)	(3,3%)
CIFAR50	75%	*Re-Play it Straight*	$3122_{\pm 351}$	$80\%_{\pm 2,2}$	$0,8_{\pm 0,021}$	$0,8_{\pm 0,022}$	$0,8_{\pm 0,022}$
			(11,2%)	(2,7%)	(2,6%)	(2,7%)	(2,8%)
		RS2	4800	$67\%_{\pm 1,9}$	$0,7_{\pm 0,018}$	$0,7_{\pm 0,019}$	$0,7_{\pm 0,018}$
				(2,8%)	(2,7%)	(2,8%)	(2,8%)
Imagewoof	65%	*Re-Play it Straight*	$2477_{\pm 243}$	$64\%_{\pm 3,7}$	$0,6_{\pm 0,038}$	$0,6_{\pm 0,037}$	$0,6_{\pm 0,037}$
			(9,8%)	(5,7%)	(5,9%)	(5,7%)	(5,9%)
		RS2	3015	$46\%_{\pm 2,6}$	$0,5_{\pm 0,028}$	$0,5_{\pm 0,026}$	$0,5_{\pm 0,027}$
				(5,7%)	(6%)	(5,7%)	(6%)

The table presents the average number of backward steps (n. steps) for our method (mean ± standard deviation, along with percentage variation), the fixed value for *RS2*, and performance metrics (Accuracy, Precision, Recall, F1-score) with their corresponding standard deviations and relative percentage variability

4.2.4 Effectiveness evaluation

To evaluate the effectiveness of our proposed technique, we define the following metric:

$$E = \frac{A_t}{s} \cdot s_{min} \tag{2}$$

where:

- A_t: is target accuracy;
- s: is number of backward steps (as introduced in Section 4.2.2) required to reach the target accuracy;
- s_{min}: is minimum number of forward steps (as introduced in Section 4.2.2) required to perform a full scan of the dataset.

This metric reaches 100% when the model achieves the target accuracy using only the minimum number of steps needed to process the entire dataset (one forward pass), indicating maximum efficiency.

As shown in Table 3, our proposed method *Re-Play it Straight* consistently outperforms *Play it Straight* across all evaluated scenarios. It also surpasses *RS2* in the cases where *RS2* outperforms *Play it Straight* (e.g., CIFAR10 and CIFAR100), except in one case in CIFAR100 where the *RS2* algorithm slightly outperforms our method, as previously discussed, due to the difficulty of the dataset. In some situations, such as the Imagewoof dataset, performance is comparable because the *Reduction of Training Data per Round* feature is not enabled due the threshold to enable it was not reached.

Furthermore, while *RS2* may show slightly better performance at lower accuracy levels in a few cases, it generally underperforms compared to our method. This is particularly evident in the CIFAR20 and CIFAR50 datasets, where *RS2* achieves higher effectiveness only with a 10% reduction factor. However, it still fails to reach the higher accuracy levels that our approach consistently attains.

4.2.5 Statistical validation

To assess the generalization ability of *Re-Play it Straight*, we conducted a statistical validation using a five-fold cross-validation approach. In this validation experiments, we divided the dataset at hand into five equal parts: in each run, the algorithm was provided with the instances in the union of four of these parts (by specifically assigning the arguments *D* and *V* of the algorithm two subsets of this union set corresponding to a 80%-20% random split of it) and tested on the remaining part. This process was repeated five times, rotating the test set in each iteration. The final performance values reported in the Table 4 are computed on the test set after training with each corresponding fold.

We set the target accuracy to the highest value obtained in our previous experiments and we took the best parameters' combination for each technique as shown in Table 3. For each dataset, we recorded the number of backward steps, as well as the accuracy, precision, recall, and F1 score – along with their standard deviations and relative standard deviations.

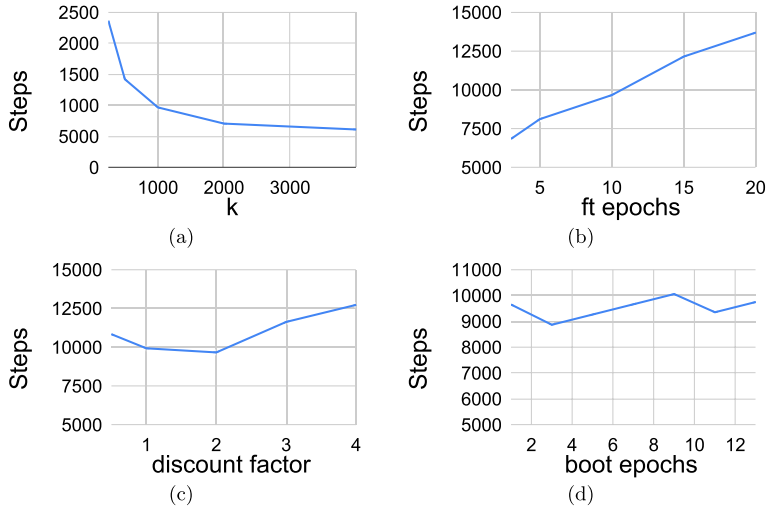


Fig. 5 These plots shows how the training steps in *Re-Play it Straight* can be influenced by the variation of the parameters: *k* in (**a**), *fine tuning epochs* per cycle in (**b**), *discount factor* in (**c**) and *boot epochs* in (**d**). These experiments were carried out on the CIFAR10 dataset

As shown in Table 4, referring to *Re-Play it Straight* 's results, the fluctuation in the number of backward steps is approximately 12%. This indicates that, even in the worst-case scenario, *Re-Play it Straight* outperforms the *RS2* algorithm. The other evaluation metrics also exhibit low variance, and the results demonstrate that the method consistently achieves values very close to the target accuracy (calculated on the test set after the validation phase). In several cases, the achieved accuracy even exceeds the predefined target.

As highlighted in the Table 4, the number of backward steps of *RS2*, being fixed due to the algorithm's design, is higher than that of *Re-Play it Straight*. This, referring to *RS2* algorithm's results, value leads to achieving the target accuracy in only one case (specifically, the CIFAR10 dataset), in all other cases, it could be more than 15% lower than the target accuracy. Notably, this metric, along with others, exhibits low variance. While these results differ from the previous experiments, the discrepancy is explained by the reduced amount of training data used in this setting (as we now apply a five-fold cross-validation approach). This demonstrates that our proposed method is not only more energy-efficient but also more effective when training data is limited.

These findings highlight the robustness and statistical consistency of the proposed method under systematic dataset variation.

4.2.6 Hyperparameters' sensitivity analysis

To better estimate the optimal set of hyperparameters for training with *Re-Play it Straight*, we conducted a sensitivity analysis by varying key parameters and plotting their trends (Fig. 5). This empirical approach provides a basis for informed and deliberate hyperparameter selection. In these experiments, all parameters were fixed as in the previous setups, by varying individually for these analysis: *k*, the number of *epochs per round*, the *discount factor*,

and the number of *boot epochs*. All tests were carried out on the CIFAR10 dataset, with a target accuracy of 90%, consistent with the benchmark used in earlier evaluations.

Figure 5a shows how performance varies with the k parameter. The curve follows a "$-\log(x)$-like" trend, indicating diminishing returns as k increases. Therefore, an effective value can be selected just after the steepest part of the curve (beyond approximately 2000 samples).

Figure 5b presents the effect of varying the number of epochs per AL-like round. The trend is monotonically increasing, suggesting that smaller values are preferable. This observation aligns with intuition: larger values risk overfitting on the small initial training data available during the early rounds.

In Fig. 5c, we analyze the impact of the *discount factor*. The curve decreases around values between 1 and 2, implying that both very high and very low values increase training costs. Specifically, a high discount factor results in minimal data usage during the boot phase, hindering the effectiveness of AL-like selection. Conversely, a very low value causes oversampling during the boot phase (values below 1), making this stage unnecessarily expensive and ineffective.

Finally, we examined the effect of the number of *boot epochs*. While this parameter is important for the fast convergence of *Re-Play it Straight*, we observed that moderate variations in its value do not significantly impact the training cost.

These experiments demonstrate the robustness and effectiveness of our method. With a more accurate hyperparameter selection based on empirical trends, we can achieve improved performance compared to the results presented in Sect. 4.2.3.

4.2.7 On the effect of adopting a cyclic learning rate mechanism in Re-Play it Straight

This section provides a detailed analysis of the behavior of *Re-Play it Straight* in terms of loss and accuracy on the CIFAR-10. The objective of this analysis is to assess not only the training dynamics of the algorithm but also its ability to generalize. Specifically, Fig. 7 illustrate the evolution of loss and accuracy at the end of each training step, reported with respect to the training subset used in that step as well as the validation and test sets. This comparison makes it possible to evaluate both the stability of the optimization process and the robustness of the learned models across datasets of increasing difficulty.

The plots highlight how both loss and accuracy undergo abrupt changes during the initial stages of each *training cycle* (Lines $7-26$ of Algorithm 2). These variations are particularly pronounced in the case of the training set, which differs at each cycle, while they appear less marked on the validation and test sets.

This behavior is typical when employing approaches such as cyclic learning rate schedules or snapshot ensembles. We argue that such learning dynamics are crucial for maximizing predictive accuracy while maintaining low energy consumption. In the following, we investigate the extent to which this behavior is attributable to the use of cyclic learning rates versus the contribution of Active Learning-based training data selection. To this end, we consider two variants of *Re-Play it Straight*. The first, referred to as LRD, preserves the use of cyclic learning rates while progressively decreasing the initial learning rate at the beginning of each cycle. The second variant, called GLRD, uses a non cyclical (global) learning rate decay.

The LRD variant still adopts a cyclic learning rate schedule with cosine annealing in each training cycle, but at the beginning of each training cycle, the initial (maximum) learning rate is set to a lower value than in the previous training cycle. Specifically, at the start of each training cycle, the learning rate is set to 80% of the initial value used in the preceding training cycle. The GLRD variant adjusts the learning rate by linearly decreasing it by 20% at each training cycle.

The empirical results of this comparison for *Re-Play it Straight* and the LRD and GLRD variants on the CIFAR-10 dataset are presented in Fig. 7, respectively (we use for *Re-Play it Straight* and all the variants the Least Confidence selection criteria, as it performed best in our previous experiments).

It can be observed from Fig. 7 that, after the initial boot phase, the trends of accuracy and loss exhibit a sawtooth-like pattern. Each of these sawtooth segments corresponds to the steps performed within a training cycle (Lines 7-26 of Algorithm 2), where the final point of the cycle corresponds to the lowest loss and highest accuracy achieved in that cycle. Hence. from the accuracy plots over the validation (*e*) and test (*f*) sets, it turns out that the use of a cyclic learning rate as proposed in *Re-Play it Straight* is essential to achieve the desired target accuracy, as the accuracy values obtained at the end of each cycle for *Re-Play it Straight* are consistently higher than those of the two variants. Furthermore, it appears evident that *Re-Play it Straight* approaches the target accuracy faster than the two variants and that it generalizes better than the two variants over the test set.

This confirms that the specific way of using cyclic learning rates in our approach is essential for enabling it to reduce training energy consumption while preserving high predictive accuracy. In particular, it mitigates the risks that would arise from relying solely on an iterative AL scheme following an initial *RS2*-based warm-up phase.

4.3 Ablation study: Re-Play it Straight with and without boost epochs

We conducted an ablation study to evaluate the impact of incorporating the extra “boost" training epochs of Steps 20-21 in the training process of *Re-Play it Straight*. Our findings indicate that the boost plays a significant role during training, particularly as the model

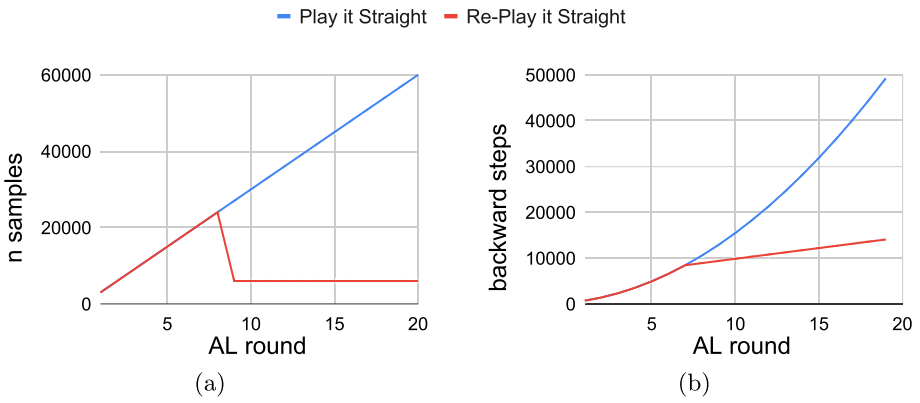


Fig. 6 These plots compare the number of samples used in each AL round (**a**) and the number of backward steps taken across different AL rounds (**b**) for *Play it Straight* and *Re-Play it Straight*, respectively. The analysis is conducted on the CIFAR-20 dataset

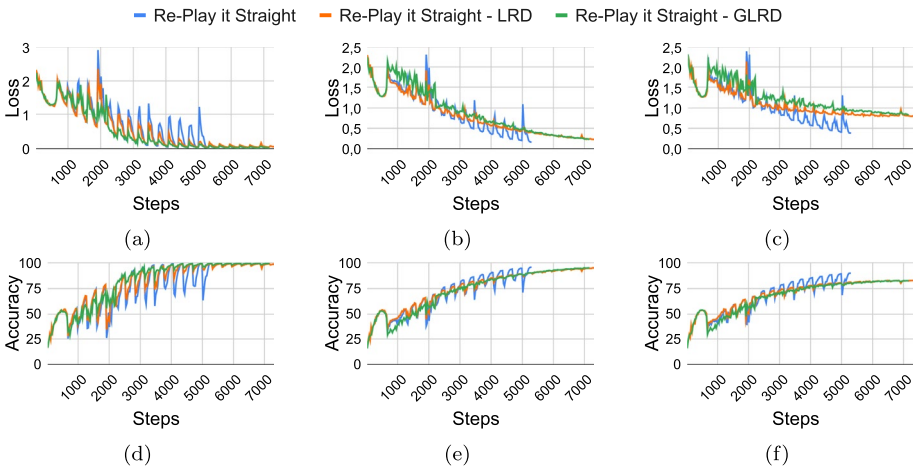


Fig. 7 Comparison between the standard *Re-Play it Straight* configuration and its Learning Rate Decay variants (LRD and GLRD, see Sect. 4.2.7) on the CIFAR-10 dataset. The top row **a-c** presents the training loss over steps for the training, validation, and test sets, while the bottom row **d-f** reports the corresponding accuracies. The x-axis indicates training steps. These plots illustrate the distinct convergence dynamics and performance differences between standard version, LRD and GLRD variant

approaches higher levels of accuracy. As illustrated in Fig. 8a and b, the boost mechanism contributes to reducing the number of training steps required to achieve high accuracy compared to the algorithm without it. However, its impact is minimal or negligible at lower accuracy levels because the training progresses quickly to achieve such levels, and as a result, the boost mechanism is rarely activated.

Interestingly, the boost proves to be more beneficial in more challenging scenarios. For instance, as shown in Fig. 8b, at the 90% accuracy level, the boost reduces the number of training steps by approximately 50% when using AL-like metrics such as Margin and Least Confidence. In other cases, the reduction is less pronounced. The boost mechanism helps the model overcome the limitation of getting stuck in local minima, which occurs due to the extremely limited amount of data encountered during training. By enabling the model to explore the entire dataset more effectively, it facilitates escaping local minima, thereby enhancing overall performance.

5 Conclusion

5.1 Discussion: contribution relevance and implications

Based on the experimental analysis described above, the approach proposed here, encoded in the form of algorithms *Play it Straight* and its enhanced version *Re-Play it Straight*, emerges as a DNN model training solution ensuring significant computational savings compared to standard training and AL approaches without compromising model accuracy. Both algorithms demonstrate compelling performance with respect to the state-of-the-art algorithm *RS2* (recently shown to work better than existing data pruning techniques) in terms of

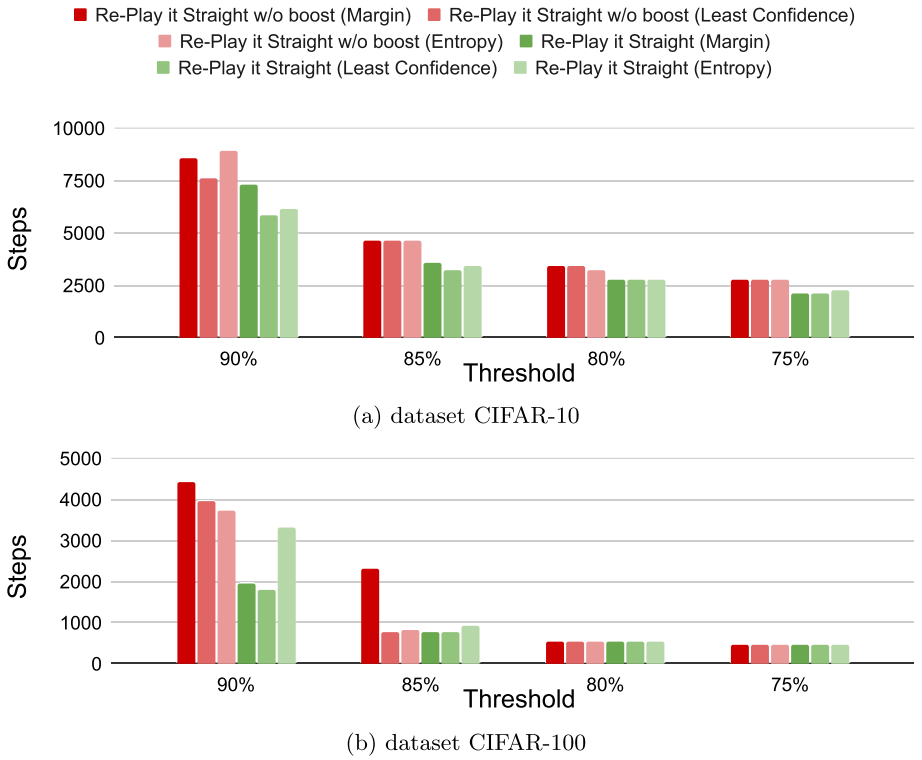


Fig. 8 Ablation study: number of steps needed to achieve the target accuracy using *Re-Play it Straight* with and without the "boost" steps 20 and 21 in Algorithm 2, using different data selection strategies (namely, Margin, Least Confidence, and Entropy) and accuracy thresholds (namely, 90%, 85%, 80%, 75% for CIFAR-10 and 75%, 70%, 65%, 60% for CIFAR-100)

the ability to minimize the computation cost and energy needed to reach a given target level of model accuracy.

Both algorithms *Play it Straight* and *Re-Play it Straight* stem from the core idea of extending an AL-like scheme, where the model at hand is trained iteratively on importance-driven data sub-samples, with additional optimization steps performed (as in algorithm *RS2* (Okanovic et al., 2024)) on uniformly-sampled data. The good performance exhibited by these algorithms allow us to empirically provide a positive answer to the fundamental research question RQ1 ("Can AL approaches to efficient DNN training be improved in a way that allow them to close the gap to the strong RS2 baseline in cost-to-accuracy performances?") that inspired our research work, and that was left as an open research direction in Okanovic et al. (2024).

In our opinion, the empirical findings presented in this work expand the state of the knowledge in the field, helping better understand the relative strengths and weaknesses of approaches relying on either random data sampling or importance-based data sampling/selection method alone (the former of which were recently shown to overcome the latter in terms of time/cost-to-accuracy (Okanovic et al., 2024; Ayed & Hayou, 2023; Guo et al., 2022)), and the advantage of adopting a hybrid approach like ours.

From a practical point of view, the good achievements of the proposed approach make us confident in the fact that it suits well resource-constrained Edge AI applications and aligns with the goals of Green AI, an increasingly important field in light of the climate crisis. Furthermore, its ability to handle large datasets expands the potential applications of deep learning models, contributing to more sustainable and affordable AI-empowered systems.

5.2 Limitations

Clearly, a major limitation of *Play it Straight* descends from the fact the volume of selected data, accumulated across different AL rounds, can grow too fast (see Fig. 6 and associated comments for an example of such a case). This exposes this algorithm to the risk of incurring high computation costs and, in extreme cases, of reaching the energy budget limit "too early", when the classification model has not been trained yet below the desired level of accuracy. This limitation has been addressed in the design of algorithm *Re-Play it Straight*, which better controls the amount of training instances per AL round, by applying a filtering mechanism to the "old" instances that have been already used in previous rounds. It is worth noting that our experimental study shows that the advantage in terms of energy saving, for a given target level of accuracy, tends to diminish when the model is to be trained at discriminating among a higher number of data classes. Further research is needed to address this issue. Anyway, both algorithms require manual setting of different hyperparameters, including the number *bootEpcs* and *ftEpcs* of optimization epochs, the reduction factor r and the number k of instance to select at each fine-tune round. Improper choices for these hyperparameters may undermine the energy saving ability of *Play it Straight* and *Re-Play it Straight*, especially if a too high number of AL rounds is required to reach the target accuracy, although in Sect. 4.2.6 we analyzed this issue and provided some practical hints to guide their selection.

Additionally, the choice of the AL-like instance ranking function is critical, as it needs to strike a balance between energy efficiency and effectiveness in data selection. If the selected function is too computationally intensive, it may vanish part of the energy savings achieved through data reduction.

5.3 Future work

To address these limitations, our future work will focus on several areas. First, we plan to investigate adaptive methods for tuning the above hyperparameters, to alleviate the burden of manual tuning and potentially improve the performances of the proposed framework across different scenarios. Second, we plan to conduct a comprehensive analysis to establish the boundaries within which *Play it Straight* and *Re-Play it Straight* demonstrates superior performance compared to other techniques. This would provide valuable guidance to practitioners and researchers in selecting the most suitable algorithm for their specific use cases. Finally, we will explore other cheap data selection strategies, combined with model-training acceleration techniques (e.g., based on model pruning, Cutout regularization, low-precision parameter quantization), to further improve *Re-Play it Straight* 's energy efficiency and effectiveness.

Acknowledgements This work was partially supported by research project FAIR (PE00000013), funded by the EU under the program NextGeneration EU.

Author contributions All the authors contributed to the conceptualization, methodology and writing of the manuscript. F.S. also performed implementation and experimental activities. All authors reviewed the manuscript.

Funding Open access funding provided by Università della Calabria within the CRUI-CARE Agreement. F.S., L.P., and partly S.F., are funded by research project FAIR (PE00000013), funded by the EU under the program NextGeneration EU

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

Code availability We make source code and evaluation data available at https://github.com/Franco7Scala/RePlayItStraight

Declarations

Conflict of interest The authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper

Ethical approval and consent to participate We declare that this research did not require Ethics approval

Consent for publication All the authors of this manuscript consent to its publication

References

Ash, J.T., Goel, S., Krishnamurthy, A., & Kakade, S.M. (2021). Gone fishing: Neural active learning with fisher embeddings. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual, pp. 8927–8939

Ayed, F., & Hayou, S. (2023). Data pruning and neural scaling laws: fundamental limitations of score-based algorithms. Trans. Mach. Learn. Res. **2023**

Bartoldson, B. R., Kailkhura, B., & Blalock, D. (2023). Compute-efficient deep learning: algorithmic trends and opportunities. *Journal of Machine Learning Research, 24*(122), 1–77.

Botev, A., Lever, G., & Barber, D. (2017). Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1899–1903 . https://doi.org/10.1109/IJCNN.2017.7966082

Chang, H.-S., Learned-Miller, E.G., & McCallum, A. (2017). Active bias: Training more accurate neural networks by emphasizing high variance samples. In: Neural Information Processing Systems . https://api.semanticscholar.org/CorpusID:20785832

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., & Zaharia, M. (2020). Selection via proxy: Efficient data selection for deep learning. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, ??? . https://openreview.net/forum?id=HJg2b0VYDr

Courty, B., Schmidt, V., Luccioni, S., Goyal-Kamal, MarionCoutarel, Feld, B., Lecourt, J., LiamConnell, Saboni, A., Inimaz, supatomic, Léval, M., Blanche, L., Cruveiller, A., ouminasara, Zhao, F., Joshi, A., Bogroff, A., Lavoreille, H., Laskaris, N., Abati, E., Blank, D., Wang, Z., Catovic, A., Alencon, M., Stechly, M., Bauer, C., & Lucas-Otavio, JPW, (2024). MinervaBooks: mlco2/codecarbon: v2.4.1. Zenodo . https://doi.org/10.5281/zenodo.11171501 .

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 . https://doi.org/10.1109/CVPR.2009.5206848

Flesca, S., Scala, F., Vocaturo, E., & Zumpano, F. (2022). On forecasting non-renewable energy production with uncertainty quantification: A case study of the italian energy market. *Expert Systems with Applications, 200*, Article 116936. https://doi.org/10.1016/j.eswa.2022.116936

Garcia-Martin, E., Rodrigues, C. F., Riley, G., & Grahn, H. (2019). Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing, 134*, 75–88. https://doi.org/10.1016/j.jpdc.2019.07.007

Goujaud, B., Scieur, D., Dieuleveut, A., Taylor, A.B., & Pedregosa, F. (2022). Super-acceleration with cyclical step-sizes. In: Camps-Valls, G., Ruiz, F.J.R., Valera, I. (eds.) Proceedings of The 25th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 151, pp. 3028–3065. PMLR, ??? . https://proceedings.mlr.press/v151/goujaud22a.html

Guo, C., Zhao, B., & Bai, Y. (2022). Deepcore: A comprehensive library for coreset selection in deep learning. In: Database and Expert Systems Applications: 33rd International Conference, DEXA 2022, Vienna, Austria, August 22–24, 2022, Proceedings, Part I, pp. 181–195. Springer, Berlin, Heidelberg . https://doi.org/10.1007/978-3-031-12423-5_14

Haimovich, D., Karamshuk, D., Linder, F., Tax, N., & Vojnovic, M. (2024). On the convergence of loss and uncertainty-based active learning algorithms. In: Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J.M., Zhang, C. (eds.) Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 . https://doi.org/10.1109/CVPR.2016.90

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., & Weinberger, K.Q. (2017). Snapshot ensembles: Train 1, get M for free. CoRR **abs/1704.00109** arXiv:1704.00109

Iyer, R., Khargoankar, N., Bilmes, J., & Asanani, H. (2021). Submodular combinatorial information measures with applications in machine learning. In: Feldman, V., Ligett, K., Sabato, S. (eds.) Proceedings of the 32nd International Conference on Algorithmic Learning Theory. Proceedings of Machine Learning Research, vol. 132, pp. 722–754. PMLR, ??? . https://proceedings.mlr.press/v132/iyer21a.html

Iyer, R.K., Khargoankar, N., Bilmes, J.A., & Asanani, H. (2021). Submodular combinatorial information measures with applications in machine learning. In: Feldman, V., Ligett, K., Sabato, S. (eds.) Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide. Proceedings of Machine Learning Research, vol. 132, pp. 722–754. PMLR, ??? . http://proceedings.mlr.press/v132/iyer21a.html

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D.P., & Wilson, A.G. (2018). Averaging weights leads to wider optima and better generalization. In: Globerson, A., Silva, R. (eds.) Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018, pp. 876–885. AUAI Press, ??? . http://auai.org/uai2018/proceedings/papers/313.pdf

Katharopoulos, A., & Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 2525–2534. PMLR, ??? . https://proceedings.mlr.press/v80/katharopoulos18a.html

Krizhevsky, A., Nair, V., & Hinton, G. (2009) Cifar-10 (canadian institute for advanced research)

Loo, N., Hasani, R., Amini, A., & Rus, D. (2022). Efficient dataset distillation using random feature approximation. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems . https://openreview.net/forum?id=h8Bd7Gm3muB

Loshchilov, I., & Hutter, F. (2017). SGDR: stochastic gradient descent with warm restarts. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, ??? . https://openreview.net/forum?id=Skq89Scxx

Mirzasoleiman, B., Bilmes, J., & Leskovec, J. (2020). Coresets for data-efficient training of machine learning models. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 6950–6960. PMLR, ??? . https://proceedings.mlr.press/v119/mirzasoleiman20a.html

Okanovic, P., Waleffe, R., Mageirakos, V., Nikolakakis, K., Karbasi, A., Kalogerias, D., Gürel, N.M., & Rekatsinas, T. (2024). Repeated random sampling for minimizing the time-to-accuracy of learning. In: The Twelfth International Conference on Learning Representations . https://openreview.net/forum?id=JnRStoIuTe

Park, D., Papailiopoulos, D., & Lee, K. (2022). Active learning is a strong baseline for data subset selection. In: Has It Trained Yet? NeurIPS 2022 Workshop

Patterson, D.A., Gonzalez, J., Le, Q.V., Liang, C., Munguía, L.-M., Rothchild, D., So, D.R., Texier, M., & Dean, J.(2021). Carbon emissions and large neural network training. ArXiv **abs/2104.10350**

Paul, M., Ganguli, S., & Dziugaite, G.K. (2021). Deep learning on a data diet: Finding important examples early in training. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems . https://openreview.net/forum?id=Uj7pF-D-YvT

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), pp. 1289–1294 . https://doi.org/10.1109/ICDM58522.2023.00163

Sachdeva, N., Wu, C.-J., & McAuley, J. (2021). SVP-CF: Selection via Proxy for Collaborative Filtering Data . https://arxiv.org/abs/2107.04984

Sachdeva, N., McAuley, J.J.: Data distillation: A survey. CoRR abs/2301.04272(2023) https://doi.org/10.48550/ARXIV.2301.04272 2301.04272

Salehi, S., & Schmeink, A. (2023). Is active learning green? an empirical study. In: 2023 IEEE International Conference on Big Data (BigData), pp. 3823–3829. IEEE Computer Society, Los Alamitos, CA, USA . https://doi.org/10.1109/BigData59044.2023.10386411

Scala, F., Flesca, S., & Pontieri, L. (2025). Play it straight: an intelligent data pruning technique for green-ai. In D. Pedreschi, A. Monreale, R. Guidotti, R. Pellungrini, & F. Naretto (Eds.), *Discovery Science* (pp. 69–85). Cham: Springer.

Scala, F., Flesca, S., & Pontieri, L. (2024). Data filtering for a sustainable model training. In: Proceedings of the 32nd Symposium of Advanced Database Systems, Villasimius, Italy, June 23rd to 26th, 2024. CEUR Workshop Proceedings, vol. 3741, pp. 205–216. CEUR-WS.org, ??? . https://ceur-ws.org/Vol-3741/paper26.pdf

Sener, O., Savarese, S. (2017). Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489

Settles, B. (2009) Active Learning Literature Survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences

Smith, L.N. (2017). Cyclical learning rates for training neural networks. In: 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017, pp. 464–472. IEEE Computer Society, ??? . https://doi.org/10.1109/WACV.2017.58 .

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In: Korhonen, A., Traum, D.R., Màrquez, L. (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pp. 3645–3650. Association for Computational Linguistics, ??? . https://doi.org/10.18653/V1/P19-1355 .

de Vries, A. (2023). The growing energy footprint of artificial intelligence. *Joule, 7*(10), 2191–2194. https://doi.org/10.1016/j.joule.2023.09.004

Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H.,Wang, X., You, Y.: Cafe learning to condense dataset by aligning features. In:Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and PatternRecognition (CVPR), United States, pp. 12186–12195 (2022). https://doi.org/10.1109/CVPR52688.2022.01188 . https://cvpr2022.thecvf.com/

Xu, J., Zhou, W., Fu, Z., Zhou, H., & Li, L.(2021). A survey on green deep learning. ArXiv **abs/2111.05193**

Yang, J., Tan, J., Wang, H., Chen, G., Wu, S., & Zhao, J. (2025). Shift guided active learning. Mach. Learn. **114**(2) https://doi.org/10.1007/s10994-024-06684-y

Yang, Z., Yang, H., Majumder, S., Cardoso, J., & Gallego, G. (2024). Data pruning can do more: A comprehensive data pruning approach for object re-identification. Transactions on Machine Learning Research

Yu, R., Liu, S., & Wang, X. (2024). Dataset distillation: a comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 46*(01), 150–170. https://doi.org/10.1109/TPAMI.2023.3323376

Zhao, B., Mopuri, K.R., & Bilen, H.(2021). Dataset condensation with gradient matching. In: International Conference on Learning Representations . https://openreview.net/forum?id=mSAKhLYLSsl