Randomness is the Root of All Evil: More Reliable Evaluation of Deep Active Learning

Yilin Ji, Daniel Kaestner, Oliver Wirth, and Christian Wressnegger
KASTEL Security Research Labs, Karlsruhe Institute of Technology
https://intellisec.de/research/eval-al

Abstract

Using deep neural networks for active learning (AL) poses significant challenges for the stability and the reproducibility of experimental results. Inconsistent settings continue to be the root causes for contradictory conclusions and in worst cases, for incorrect appraisal of methods. Our community is in search of a unified framework for exhaustive and fair evaluation of deep active learning. In this paper, we provide just such a framework, one which is built upon systematically fixing, containing and interpreting sources of randomness. We isolate different influence factors, such as neural-network initialization or hardware specifics, to assess their impact on the learning performance. We then use our framework to analyze the effects of basic AL settings, such as the query-batch size and the use of subset selection, and different datasets on AL performance. Our findings enable us to derive specific recommendations for the reliable evaluation of deep active learning, thus helping advance the community toward a more normative evaluation of results.

1. Introduction

The availability of labeled data is crucial for effectively training deep learning methods. However, obtaining good labels is often more expensive than mere computational power, such that the research community has invested large efforts in active learning (AL) [29]. By selecting the most informative samples, the number of required *labeled* data points to train a classifier can effectively be reduced. Most commonly one selects an initial set of samples for bootstrapping the classifier ("init set") and queries samples in batches ("query batch") to retrieve additional labels. At each iteration, the classifier is trained using all labeled data, either based on the random initialization of the model's weights ("cold start") or building upon the learned weights from the previous round ("warm start"). Selection strategies can be roughly categorized in (a) uncertainty-based sampling [e.g., 9, 14, 24, 35, 39, 42], (b) diversity-based sampling [e.g., 34, 37], and (c) combined approaches [e.g., 3, 4, 8, 15, 18].

Despite recent advances, reproducibility and rigorous comparative evaluation remain major challenges in practice [19]. For instance, the random initialization of "init sets" and weights of the backbone model can have a large impact on the performance of an AL strategy. Recent research sets out to address the difficulty in evaluating deep active learning [5, 22, 25] but puts focus on individual aspects only rather than on providing an overall picture, as indicated in Table 1. In some rare cases, analyses even bring forward contradictory conclusions. As an example, Munjal et al. [25] show that AL performance is inconsistent under different sizes of query batches, whereas Beck et al. [5] conclude that the query-batch size only has a negligible effect.

Although we as a community have managed to identify the problem, we still struggle to address it adequately since the experimental setup in (deep) active learning is influenced by so many subtle yet decisive factors. Therefore, it is crucial that we establish a community-wide and consistent evaluation framework, reaching a common understanding of how to control and contain the influence factors.

In this paper, we identify factors that influence the performance of different active learning strategies in three categories: (1) the underlying learning setup, (2) different sources of randomness, and (3) specifics of the execution environment. For each of these factors, we systematically *evaluate the sensitivity on a strategy's overall performance* using tests for statistic significance inspired by Ash et al. [3, 4] and explore the underlying reasons. We discuss means to control these influence factors as part of the model's training process and *provide specific recommendations* on how each aspect can be handled in practice to yield more reproducible results. Finally, we then resume to additionally *analyze different parameters of active learning* based on our framework and substantiate crucial influence on performance under strictly controlled experimental settings.

We want to stress that this paper is not intended as a finger-pointing exercise, but as an attempt to push active learning research forward to more rigorous evaluations. Our framework provides the urgently needed tools for comprehensive and reliable comparison in this thorny and complex domain.

	Aspect	Beck et al. [5]	Lang et al. [22]	Munjal et al. [25]	Ours
LEARNING	Data Augmentation	●	●	●	●
	Optimizer	●	●	●	●
	Hyper-parameter Opt.	–	○	●	●
	Backbone Architecture	–	●	–	●
	Regularization	●	–	●	–
	Validation Set	–	–	●	–
	Early Stopping	–	–	–	●
RAND	Initial Seeds/ Seed Sets	●*	●*	○	●
	Mode Initialization	–	–	–	●
	(Non-)Deterministic Comp.	–	–	–	●
	Warm-/Cold Start	●	●	–	●
ENV	Code Base	–	●	–	●
	Hardware Difference	–	–	–	●
Analyze AL	Query-Batch Size	●	–	○	●
	Subset Sampling	–	–	–	●
	Dataset Imbalance	–	–	●	●
	# Samples per Class	●	–	–	●
	Scalability	●	–	●	●
	Overall Metric	–	–	–	●

Table 1: Overview of the community's attempt to systematize evaluating active learning. Full black circles (●) mark considered aspects, full gray circles (●) indicate that community knowledge has been incorporated, and empty circles (○) mean partial analyses. The asterisk (*) marks contributions that have focused on ways of constructing initial sets rather than randomness.

2. Reliable Evaluation of Deep Active Learning

In order to reliably evaluate learning-based approaches, we have to carefully prepare the experimental setup to contain randomness. As a results of a multitude of different influence factors, active learning is a particularly difficult application in this regard. In the following, we present a framework for limiting sources of randomness and demonstrate their influence on detection performance.

We begin by outlining the five exemplary active learning methods that we use for demonstrating the impact of our framework's design choices, before we present the metric for showing statistical significance. In Section 2.1, we then discuss measures to control the underlying machine learning setup, before we propose specific actions for containing randomness in Section 2.2. In Section 2.3, we additionally stress the importance of fixing the hardware and software environments across comparative evaluations.

Methods under test. We re-implement 7 AL methods (BADGE [4], BALD [13], Core-Set [34], Entropy [35], ISAL [24], LC [23], and LLOSS [42]) across uncertainty-based, diversity-based, and combined strategies, and accompany them with a simple random selection strategy. Table 2 summarizes their respective setups. All experiments are conducted on the CIFAR-10 and CIFAR-100 dataset [20] for which we follow the recommendations assembled in Sections 2.1 to 2.3. Each experiment is repeated multiple times and we randomly select 1,000 data points for the initial set and retrieve 2,000 samples within a query batch, unless otherwise specified. Fig. 1 provides a first glimpse at the performance of the five considered strategies. While the label efficiency [5] is a useful metric, it is apparent that the trend up to 5–10 k is not expressive. Moreover, we observe that BALD, BADGE, Entropy, ISAL, LLOSS and LC arrive at the maximum accuracy (the accuracy of a network trained on the entire training dataset) at 20 k to 30 k labeled samples. For a conclusive evaluation, thus, comparing performance from zero to this converged point is crucial. Hence, in all subsequent experiments, we display results from 0 to 25 k labeled samples across different batches on the x-axis.

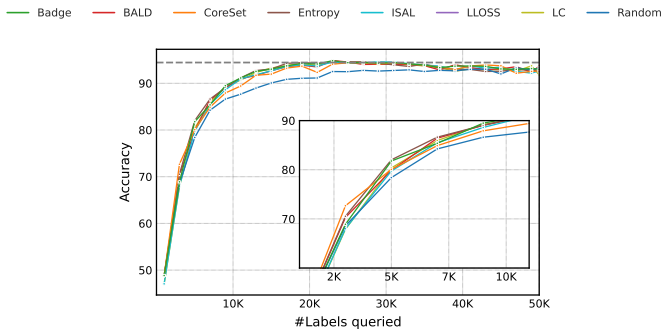


Figure 1: Overall performance of AL methods on CIFAR-10.

All experiments are conducted using NVIDIA A-100 GPUs, except for measurements performed to compare the influence of different hardware and non-deterministic training which are run on NVIDIA RTX 3090 cards.

Significance tests. As a means to compare measurements across different settings, we perform $T = 3$ trials with different initial seeds, each at various labeling budgets.

These are then statistically analyzed using two-tailed paired t-tests [30, 31]. The results are visualized as pair-wise penalty matrices (PPM) [3, 4] for which we do a pair-wise comparison of two AL methods i and j after each batch. The t-value is then calculated as $t = \frac{\sqrt{T}\hat{\mu}}{\hat{\sigma}}$ with

$$\hat{\mu} = \frac{1}{T}\sum_{k=1}^{T}(a_k^{(i)} - a_k^{(j)}) \quad \hat{\sigma} = \sqrt{\frac{1}{T-1}\sum_{k=1}^{T}(a_k^{(j)} - a_k^{(j)} - \hat{\mu})}$$

where $\{a_1^{(i)}, \ldots, a_T^{(i)}\}$ and $\{a_1^{(j)}, \ldots, a_T^{(j)}\}$ denote the accuracy for T trials of method i and j.

Considering a confidence level of $90\,\%$, we yield a t-value interval of $[-2.92, 2.92]$ where two methods i and j are considered as *not* significantly different. If the t-value is above that interval (method i outperforms method j) or below it (method j outperforms method i), we add a penalty score of $\frac{1}{n}$ to the cell of the pair-wise penalty matrix at (i, j) or (j, i), respectively. Here, n is the total number of measuring points (the number of batches) over the increasing number of labelled data for the current experiment. The higher the value is in the resulting heat map, the stronger the method at row i dominates the method associated with column j. Consequently, the method corresponding to the column with the lowest values performs best and the one with the highest values the worst. In order to compose rankings, we use the column-wise average marked as $\varnothing$. As an example (that we discuss in more detail in Section 2.2), Fig. 2 shows a pair-wise penalty matrix for different model initializations.

2.1. Controlling the Underlying Learning Setup

Recent research shows that the configuration of the underlying learning setup, that is the training settings of the target model, has impact on evaluation results [5, 22]. While these aspects are not our paper's main focus, we briefly survey crucial design choices to derive specific recommendations.

2.1.1 Backbone Architecture

In particular for task-ware active learning methods (e.g., Core-Set or BADGE) the underlying learner has a decisive impact as the features derived from it are used as input for the sample selection process. Lang et al. [22] investigate the effect of the backbone architectures and compare AL methods on CIFAR-10 learned with VGG16, ResNet18, and DenseNet201. They conclude that (a) ResNet18 can achieve the highest accuracy and that (b) using an architecture incompatible with the dataset (e.g., DenseNet201 for CIFAR-10) can reduce performance of active learning significantly. Additionally, it is of the utmost importance that the community agrees upon a common definition of the individual backbones, so that, for instance, ResNet18 of a specific evaluation is identical with the used ResNet18 of another work. In Appendix B, we detail and summarize network definitions across active learning research.

Recommendation. Use the backbone architecture with the community-accepted definition that is best suited for the dataset at hand and consistently use it across all experiments. In the image classification domain, we suggest using ResNet18 for CIFAR-10 and CIFAR-100.

2.1.2 Types of Optimizer

Compared to stochastic gradient descent (SGD) [43], adaptive optimizers such as Adam [16] and RMSProp [12] show poor generalization despite faster convergence [40, 44]. Hence, the choice of optimizer can be crucial for assessing the final performance. Beck et al. [5] and Lang et al. [22] conduct experiments to investigate the effect of using SGD and Adam optimizer, concluding that SGD results in higher label efficiency with the same backbone and hyperparameters on the same dataset. The significance of this is further underlined by the overview provided in Table 2, showing a diverse use of optimizers.

Recommendation. Control the type of optimizer across methods for comparative evaluations to ensure that the yield performance difference stems from an active learning method itself. As SGD often generalizes better, we encourage its use for deep active learning.

2.1.3 Learning Rate

In addition to a suitable backbone architecture (with a community accepted structure), the training's hyperparameters need to be selected carefully. For instance, the most suitable learning rate depends on the dataset, optimizer, and not least the backbone architecture itself. Lang et al. [22] shows that for CIFAR-10 with SGD, a larger learning rate is beneficial, while Munjal et al. [25] even suggest that instead of fixing the hyperparameters upfront for all AL iterations, they may be tuned at each step using AutoML. While we explicitly acknowledge the importance of hyperparameter tuning, a continuous adaptation is time-consuming.

Recommendation. Pragmatically fix the learning rate to 0.1 for SGD on image datasets. While continuous hyperparameter tuning can improve overall performance, a fixed learning rate does not change the ranking of AL methods from a comparative evaluation's point of view.

2.1.4 Data Augmentation

While data augmentation is popular in deep learning as a means to address overfitting, its significance for active learning is often neglected. Beck et al. [5] show that overall accuracy and label efficiency can be improved with data augmentation. As an example, BADGE [4] achieves 10 percentage

points higher accuracy towards the end of the active-learning cycles compared to results without data augmentation, which Lang et al. [22] have confirmed this finding. Beside the overall improvement of classification performance, they also point out that data augmentation can affect the ranking of active-learning methods if used inconsistently. In contrast to adaptive hyper-parameter tuning, data augmentation can be incorporated at comparably small training-time costs. Hence, a widespread use can be accepted more easily in practice.

Recommendation. One may use data augmentation if applied consistently across methods, such that it does not affect the overall ranking. However, a commonly accepted baseline is needed, e.g., random horizontal flipping and random cropping for image classification.

2.1.5 Early Stopping

Yoo and Kweon [42] have identified 200 epochs as a practical setting for training ResNet18 on CIFAR-10, when the model is fully trained but not overfitting. While this setting is widely used in the AL community [8, 15, 42], using early stopping or a fixed number of epochs can have a impact on the evaluation as we present in Appendix D.

2.2. Containing Randomness

Perhaps the most obvious influence factor on experimental design is randomness. While the need for controlling randomness is non-controversial [4, 25], the multitude of manifestations is difficult to oversee. A common way of handling randomness in evaluations is to repeat each experiment several times and report averaged results with their standard deviation. However, if the fluctuation of a specific approach's performance is larger than the improvement over its contestants, results are difficult to interpret. Munjal et al. [25] perform statistical analyses of AL results on a macro level for parameter initialization and data augmentation (cf. Table 1), and provide first valuable insights.

In this section, we extend upon this result and set out to unravel different sources of randomness. We discuss different aspects to the problem and analyze their influence on AL performance using the example of BADGE [4], BALD [13], Core-Set [34], Entropy [35], ISAL [24], LC [23], and LLOSS [42]) and the random strategy.

2.2.1 Model and Method Initialization

In the absence of a suitable alternative, the learning setup is most commonly bootstrapped with random initialization or with an initialization scheme that involves randomness to a certain extend. For active learning, we have multiple such scenarios, for instance, for (a) initializing the backbone model and (b) initializing the active-learning method itself. Moreover, the learned model is commonly reinitialized after new samples are queried for labeling ("cold starts"), increasing the introduced randomness with every batch.

Key to controlling randomness is to update the backbone model as new samples are queried. Instead of learning the model from scratch with new random initialization, the model is initialized with the parameters from the previous round ("warm starts"). Consequently, all remaining randomness stems from the initialization in the first round of active learning, which already stabilizes the comparative measurements significantly. For reliable comparative evaluation, initialization is fed with fixed inputs over multiple runs to average out the remaining randomness. This forms a sequence of R tuples, $(s_1, \ldots, s_l)$, containing seeds for initializing the individual factors for one specific run. As active learning is runtime expensive, the number of repetitions is usually relatively small in practice. It thus is crucial that all methods under investigation receive the same tuple of random seeds to establish consistency across experiments.

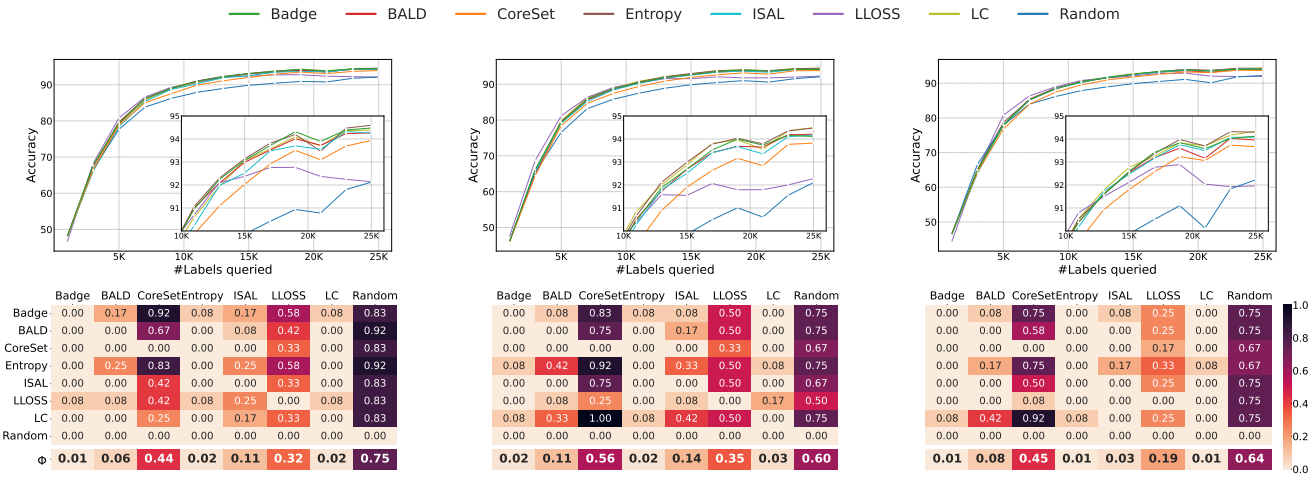


Figure 2: Analysis of active learning performance for three different init sets.

More specifically, we initialize T "init sets" using fixed random seeds and use each of them to train models with T different weights initialization, resulting in $R = T \times T$ models and thus trials. Moreover, it is important to note that different execution environments have different random number generators to provide randomness. As an example, we control init sets in our experiments by specifying seeds through Python and weight initialization using PyTorch.

Recommendation. Refine model parameters across AL batch ("warm starts") to prevent exhaustive reinitialization and feed initialization of the backbone model's weights and the "init sets" with fixed inputs over multiple runs to average out the randomness. Moreover, use identical seeds for all methods under investigation.

In order to reveal the influence of these three different factors, we statistically analyze their impact on active learning performance. We apply all recommendations from Section 2.1 and Section 2.2 so far and actively learn on CIFAR-10 and CIFAR-100 with eight different strategies. Additionally, we ensure fully-deterministic computations to further reduce potential side-effects. Recall, that we initialize our experiments based on $T \times T$ tuples of initialization seeds, $\{(I_1, M_1), \ldots, (I_1, M_T), (I_2, M_1), \ldots (I_T, M_T)\}$. The various influences are then measured using paired t-tests over different groupings of these tuples.

Influence of initialization sets ("init sets"). To determine the influence of differently initialized init sets, we conduct experiments as described above. We perform t-tests for models that use the same init set seed, $\{(I_k, \star)\}$, to determine the best performing strategy represented as pair-wise penalty matrices. Fig. 2 shows the result of three out of T such groups (cf. previous page), with the accuracy for multiple active learning strategies on the y-axis over an increasing amount of labeled data on the x-axis at the top, and the corresponding pair-wise penalty matrices at the bottom.

Using the first initial seed set (left), the column representing BADGE has the lowest average penalty-score ($\varnothing$ 0.01), that is, BADGE outperforms the other strategies, but is closely followed by Entropy ($\varnothing$ 0.02) and LC ($\varnothing$ 0.02). For the second (middle) and third seed (right), BADGE performs similar to Entropy. Moreover, BALD ($\varnothing$ 0.08) falls behind ISAL ($\varnothing$ 0.03) with the third seed while BALD beats ISAL with other two seeds. The accuracy progressions also suggest consistent observation. Interestingly, comparing with the first and second seeds, the performance of LLOSS ($\varnothing$ 0.19) using the third seed significantly fluctuates. Similar analysis on CIFAR-100 are presented in Appendix E. Thus, we can conclude that seeds can change method rankings (especially for tight calls), which is consistent with qualitative observations from Munjal et al. [25].

Influence of model initialization. Next, we perform t-tests for models that use the same seeds for initializing the model's weights, $\{(\star, M_k)\}$, to compare the performance of active learning strategies. In Fig. 3, we show result of three out of T performance progressions and the corresponding pair-wise penalty matrices underneath it. For the first model initialization (left) and the third model initialization (right), Entropy ($\varnothing$ 0.01) outperforms BADGE ($\varnothing$ 0.04) slightly. However, for the second seed (middle), BADGE ($\varnothing$ 0.02) clearly outperforms Entropy ($\varnothing$ 0.06). For ISAL and LLOSS, we observe that the penalties vary strongly across seeds, indicating that this method is particularly prone to variance of the model initialization. We yield similar results on CIFAR-100

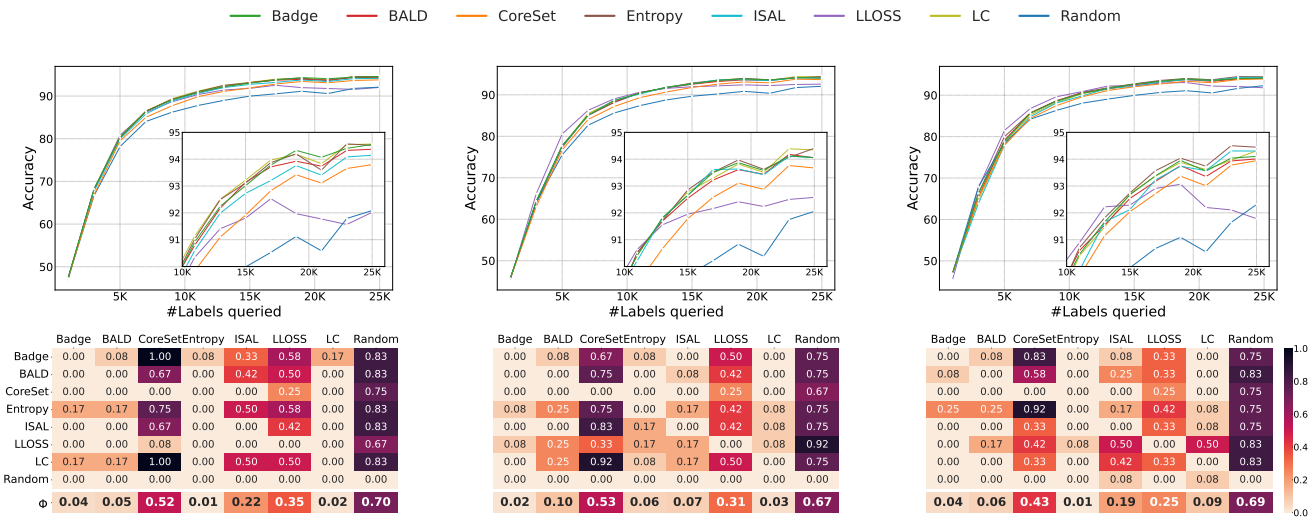


Figure 3: Analysis of active learning performance for three different model initialization.

that we present in Appendix E. In summary, we observe that (a) active learning strategies expose vastly different degrees of change for varying seeds and (b) Entropy, BALD, and BADGE seem more robust to varying initialization compared to Core-Set, LLOSS and ISAL.

Influence of updating model weights ("warm starts"). While we are the first to discuss the use of "warm starts" for stabilizing active learning for evaluation purposes, the fact that it yields different performance than repeated "cold starts" has been investigated in prior work [5, 22]. In our experiments that we report in Appendix C, we confirm this in our setting as well.

2.2.2 Computation

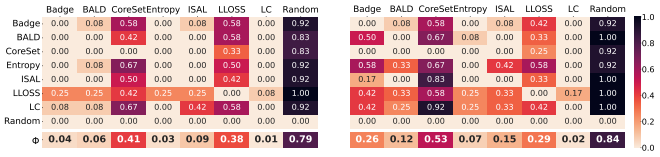
In addition to randomness that is made explicit by setting seed values as discussed in the previous section, also the computation platforms themselves can be subject to randomness at their very core. For instance, CUDA's convolution operation is non-deterministic per default [45] and the developer has to explicitly request deterministic computation [28].

Usually, non-determinism of fundamental operations can be easily compensated for by averaging over multiple runs. For active learning, however, recent research yield rather small improvements over related work. Consequently, the stochastic variance of non-deterministic operations can influence these results, rendering the explicit use of deterministic computations necessary. Although this significantly increases computation time, we perform all our prior and subsequent experiments with deterministic computations to avoid any interference.

Recommendation. Run experiments multiple times to compensate for non-deterministic operations. If the resulting variance is larger than the gained improvement, use deterministic operations stringently.

Influence of non-deterministic training. Once more, we run experiments with all eight active learning strategies, following our previous recommendations. For each strategy, we run T trials, each consisting of one deterministic run and three non-deterministic runs that we average. Both groups are compared using pair-wise t-tests anchored at the seed tuple, (I_k, M_k), defining the initialization for a single run.

Fig. 4 shows pair-wise penalty matrices for non-deterministic training (left) and deterministic training (right). In Appendix F, we additionally provide a visualization of the progression of accuracy values. At first sight it is apparent that the heatmap's pattern differs unmistakably. While BADGE ($\varnothing$ 0.04) and Entropy ($\varnothing$ 0.03) perform similarly for non-deterministic computations, for the deterministic setting, in turn, BADGE ($\varnothing$ 0.26) significantly falls behind



(a) Non-deterministic (b) Deterministic

Figure 4: Analysis of (non-)deterministic computations.

Entropy ($\varnothing$ 0.07) and BALD ($\varnothing$ 0.12). Also LLOSS elevates from $\varnothing$ 0.38 to $\varnothing$ 0.29, while Core-Set deteriorates from $\varnothing$ 0.41 to $\varnothing$ 0.53, further underlining the relevance of deterministic training.

2.3. Fixing the Execution Environment

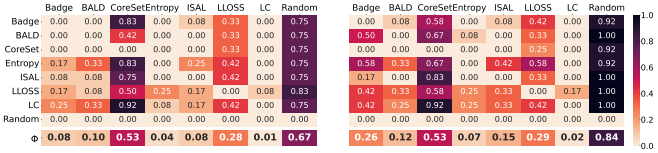
The execution environment has a potentially defining influence on active learning beyond randomness, extending towards software and hardware implementations. Oftentimes, the true nature of the experimental setup only becomes apparent in the provided open-source implementation of a particular approach (if available). Consequently, before using any implementation all influence factors have to be verified. For comparative evaluations, it is not sufficient to reuse implementations. Instead it is necessary to adjust source code to fix crucial parameters such as the underlying backbone architecture, the optimizer, the learning rate, or the use of data augmentation. At the same time, the specific hardware (e.g., used GPU model) can have an impact on the active-learning performance. Unfortunately, training results are not guaranteed to be comparable for different GPUs, even when using identical seeds and deterministic training.

Recommendation SW. Configure and verify influence parameter in active learning implementations thoroughly. To foster future research, we provide implementations as part of our framework at: `https://intellisec.de/research/eval-al`

Recommendation HW. Ensure that comparative evaluations are run on identical hardware. While it is not necessary to execute all experiments on the same physical device, the GPU model, for instance, should be the same. Do not mix hardware and list hardware details.

Influence of varying GPU models. All prior experiments have been conducted on consistent platforms. For evaluating the influence if this was not the case, we run T trials on two types of GPUs (NVIDIA A-100 and NVIDIA RTX 3090) with deterministic computations and compare results using pair-wise t-tests anchored at the seed tuple, (I_k, M_k), as proposed for prior experiments. The results are provided in Fig. 5 as pair-wise penalty matrices.

Again, we see that BADGE, in particular, is subject to change depending on this influence factor, ranked third (⌀ 0.08) closely behind LC (⌀ 0.01) and Entropy (⌀ 0.04) and before BALD (⌀ 0.10),LLOSS (⌀ 0.28) Core-Set (⌀ 0.53), and Random (⌀ 0.67) for NVIDIA A-100 GPUs. For NVIDIA RTX 3090 cards, in turn, BADGE (⌀ 0.26) falls behind BALD (⌀ 0.12) obviously. The results further underline the necessity for consistency of experimental setups including hardware details.



(a) NVIDIA A-100 (b) NVIDIA RTX 3090

Figure 5: Analysis of different GPU models.

3. Analyzing Active Learning

Based on the evaluation framework that we describe in the previous section, we can know analyze the impact of different active learning settings. In particular, we consider the query-batch size in Section 3.1, the usage of subset sampling in Section 3.2, and different datasets in Section 3.3. Additionally, in Section 3.4, we then perform an overall comparative evaluation of five active learning strategies.

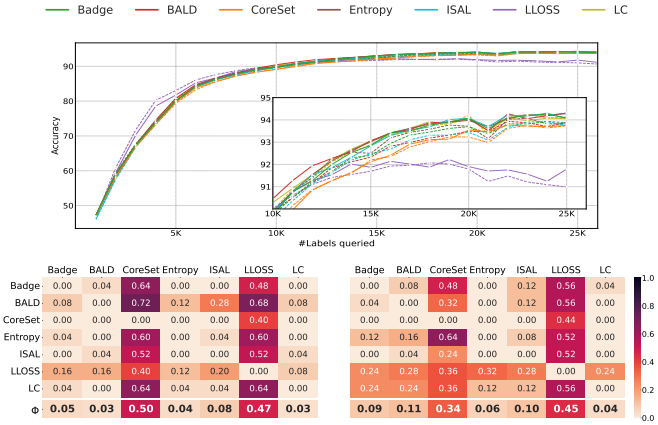
3.1. Query-Batch Size

The size of the batch of samples to be queried for labels is central to active learning. We thus fix the experimental setup as described in Section 2 and analyze the influence of the query-batch size based on three settings: 1,000, 2,000, and 4,000. We compute one pair-wise penalty matrix for each size to compare active learning strategies and report the results in Fig. 6. We observe that the ranking of AL methods slightly varies with the different query batch sizes. For a batch size of 1,000, BALD ranks first identically (⌀ 0.03), but is significantly outperformed by Entropy for batch sizes of 2,000 and 4,000, with ⌀ 0.10 to ⌀ 0.04, and ⌀ 0.21 to ⌀ 0.0, respectively. Also, BALD is even with LC at first, but falls behind for larger batch sizes. In Appendix G , we additionally provide the analysis on CIFAR-100.

Recommendation. Consider multiple query-batch sizes in the evaluation. The choice of the sizes needs to be appropriate for the total number of unlabeled samples.

3.2. Subset Sampling

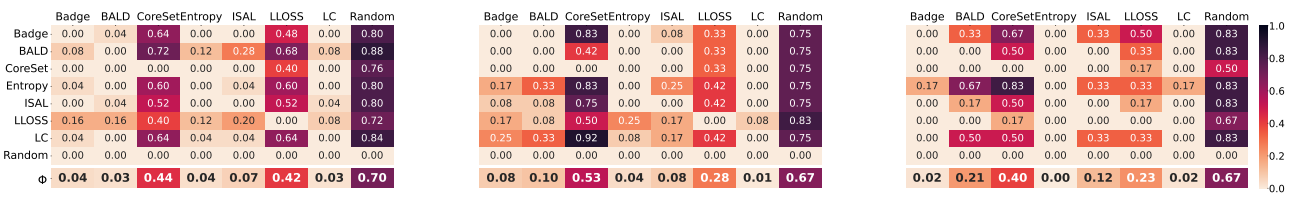
As can be seen in Table 2, multiple active learning strategies employ subset sampling in their evaluation [8, 15, 42]. The influence of sub-sampling on aparticular active learning strategy, however, often is opaque and itremains unclear how much sub-sampling contributes in comparison to the newly proposed approach. Hence, we study this complementary measure under our framework and report the results for pairwise comparisons of different active learning methods in Fig. 7. Note that the random strategy, of course, is equivalent with and without sub-sampling and is excluded here.



(a) Full (b) Sub-sampled

Figure 7: Analysis of different subsets. Performance full and sub-sampled data is indicated as solid and dashed lines.

We can observe that sub-sampling has different influence on different strategies. For BALD and BADGE the results with sub-sampling are worse than that without, while for Core-Set the influence is (relatively) smaller, such that it is able to shrink the gap to the other approaches. This tendency is also visible in the accuracy progression where all approaches but Core-Set and LLOSS show a significant difference. While for the full dataset, BALD and LC are on a par (⌀ 0.03) and slightly better than Entropy (⌀ 0.04),



(a) 1,000 samples (b) 2,000 samples (c) 4,000 samples

Figure 6: Analysis of varying batch sizes.

with sub-sampling LC ($\varnothing$ 0.04) significantly outperforms BALD ($\varnothing$ 0.11). We also yield similar observations for CIFAR-100 in Appendix H. Thus, using sub-sampling can change the ranking of AL strategies. Interestingly,[42] mentions that sub-sampling might alleviate the overlapping of selections for uncertainty-based methods. We observe that this highly depends on type of dataset. For instance, on CIFAR-100, LC and Entropy perform clearly better after using sub-sampling, while on CIFAR-10, the observation is the other way around.

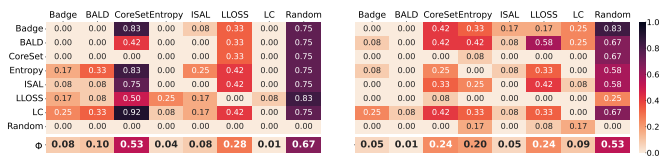
Recommendation. Compare active learning strategies without sub-sampling, unless one of the approaches uses it as a fundamental building block. In this case a detailed analysis of the influence of sub-sampling is necessary.

3.3. Datasets

Next, we investigate the relevance of evaluating AL methods on imbalanced datasets and large-scale datasets that have a larger number of classes and/or more samples per class.

Imbalanced datasets. Most datasets that are commonly used in active learning research have perfectly balanced classes. However, for assessing the practicability of active learning this cannot and must not be assumed. Kim et al. [15] already show that the ranking of AL methods differs between balanced and imbalanced datasets. We extend upon this observation in Appendix I.

Scalability to CIFAR-100 and TinyImageNet. We start to investigate active learning performance on the CIFAR-100 dataset and, thus, with $10\times$ more classes then in our initial evaluation. The pair-wise penalty matrix in Fig. 8a shows that most of the time LC dominates the ranking for CIFAR-10. On CIFAR-100, however, BALD ($\varnothing$ 0.01) surpasses all other approaches clearly, including LC (Fig. 8b). Core-Set ($\varnothing$ 0.53) falls significantly behind Entropy ($\varnothing$ 0.04) on CIFAR-10, however, their performances are close to each other on CIFAR-100. Moreover, the improvement over Random is reduced on CIFAR-100 compared to CIFAR-10, which is also corroborated by Beck et al. [5]. We also analyze the performance on the large-scale dataset TinyImageNet in Appendix J. We conclude that AL strategies have different scalability to different types of datasets.



(a) CIFAR-10 (b) CIFAR-100

Figure 8: Analysis of different datasets.

Recommendation. Evaluate active learning strategies on multiple benchmark datasets, that comprise balanced, imbalanced, small-scale, and large-scale datasets to cover most relevant cases in practice.

3.4. Comparative Analysis

Finally, we outline an overall comparative evaluation of the five exemplary active learning strategies. We compare the methods with pair-wise penalty matrices summarizing different query batch sizes, $\{1000, 2000, 4000\}$, and datasets, $\{\text{CIFAR-10}, \text{CIFAR-100}\}$. All experiments are conducted according to our framework and are analyzed across $T = 3$ trials with a confidence level of $90\,\%$.

Fig. 9 shows the corresponding pair-wise penalty matrix, summing up all experiments. The last row shows the column-wise average per active learning strategy that allows to concisely derive a ranking. The lower the column-wise mean, the better the method in comparison to the other methods. Surprisingly, LC ($\varnothing$ 0.34) and BADGE ($\varnothing$ 0.34) dominates the rankings, followed by BALD ($\varnothing$ 0.44),ISAL ($\varnothing$ 0.53), Entropy ($\varnothing$ 0.72), LLOSS ($\varnothing$ 1.45) and Core-Set ($\varnothing$ 2.18).

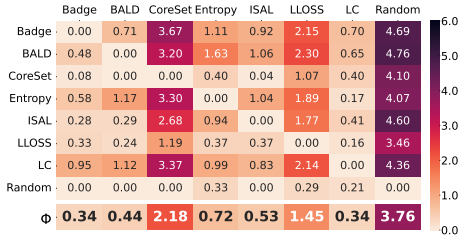


Figure 9: PPM over all experiments

Recommendation. For a comprehensive analysis of AL strategies, the overall comparative evaluation should incorporate as many variables from Section 3 to yield a summarized PPM that is as expressive as possible.

4. Conclusion

Recently, our community has identified critical shortcomings in reproducing experimental results of active learning. Despite having acknowledged the problem, so far, we have been lacking a comprehensive framework for reliable evaluation of novel approaches. In this paper, we fill this gap by systematically fixing, containing, and interpreting sources of randomness. We provide specific recommendations for the research practitioner that help set up active learning experiments. We thus provide urgently needed tools for comprehensive and reliable evaluation in this challenging domain.

Acknowledgments. The authors thank the anonymous reviewers and gratefully acknowledge funding by the Helmholtz Association (HGF) within topic “46.23 Engineering Secure Systems”.

References

[1] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. *Combinatorial and Computational Geometry*, 52, 2005.

[2] Jordan Ash. Batch active learning by diverse gradient embeddings (BADGE). `https://github.com/JordanAsh/badge/blob/master/run.py`, 2020.

[3] Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham M. Kakade. Gone fishing: Neural active learning with fisher embeddings. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[4] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020.

[5] Nathan Beck, Durga Sivasubramanian, Apurva Dani, Ganesh Ramakrishnan, and Rishabh Iyer. Effective evaluation of deep active learning on image classification tasks. *CoRR*, abs/2106.15324, 2021.

[6] Maryam Biazaran and Bahareh SeyediNezhad. *Facility Location: Concepts, Models, Algorithms and Case Studies*, chapter 9, pages 193–217. Physica Heidelberg, 2009.

[7] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The balanced accuracy and its posterior distribution. pages 3121–3124, 2010.

[8] Razvan Caramalau, Binod Bhattarai, and Tae-Kyun Kim. Sequential graph convolutional network for active learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9583–9592, 2021.

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 27, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] Geoffrey Hinton. Rmsprop: Divide the gradient by a running averageof its recent magnitude. Lecture, 2012.

[13] Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *CoRR*, abs/1112.5745, 2011.

[14] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2372–2379, 2009.

[15] Kwanyoung Kim, Dongwon Park, Kwang In Kim, and Se Young Chun. Task-aware variational adversarial active learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8166–8175, 2021.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2015.

[17] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2014.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

[19] Daniel Kottke, Adrian Calma, Denis Huseljic, Georg Krempl, and Bernhard Sick. Challenges of reliable, realistic and comparable active learning evaluation. In *Proc. of the Workshop and Tutorial on Interactive Adaptive Learning*, 2017.

[20] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR (canadian institute for advanced research).

[21] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *CoRR*, (arXiv:1910.09700), 2019.

[22] Adrian Lang, Christoph Mayer, and Radu Timofte. Best practices in pool-based active learning for image classification. *OpenReview*, 2021.

[23] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proc. of the Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. ACM/Springer, 1994.

[24] Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. Influence selection for active learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9274–9283, 2021.

[25] Prateek Munjal, Nasir Hayat, Munawar Hayat, Jamshid Sourati, and Shadab Khan. Towards robust and reproducible active learning using neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 223–232, 2022.

[26] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[27] William H. Press and Glennys R. Farrar. Recursive stratified sampling for multidimensional monte carlo integration. *Computers in Physics*, 4(190), 1990.

[28] PyTorch Contributors. Reproducibility. `https://pytorch.org/docs/stable/notes/randomness.html`, 2022.

[29] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

[30] John A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Advanced Series, 2006.

[31] Amanda Ross and Victor L. Willson. *Basic and Advanced Statistical Tests*. SensePublishers Rotterdam, 2017.

[32] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *Proc. of the European Conference on Machine Learning (ECML)*, pages 413–424, 2006.

[33] Yassir Saquil, Kwang In Kim, and Peter M. Hall. Ranking cgans: Subjective control over semantic image attributes. In *Proc. of British Machine Vision Conference (BMVC)*, 2018.

[34] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[35] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proc. of the Confer-*

ence on Empirical Methods in Natural Language Processing (EMNLP), pages 1070–1079, 2008.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Proc. of the International Conference on Learning Representations (ICLR)*, 2015.

[37] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5972–5981, 2019.

[38] Aad W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

[39] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

[40] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[41] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017.

[42] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 93–102, 2019.

[43] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proc. of the International Conference on Machine Learning (ICML)*, 2004.

[44] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, et al. Towards theoretically understanding why SGD generalizes better than ADAM in deep learning. *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 33:21285–21296, 2020.

[45] Donglin Zhuang, Xingyao Zhang, Shuaiwen Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. In *Proc. of the Proceedings of Machine Learning and Systems (MLSys)*, 2022.

Method	Backbone Model	Optimizer	LR	Data Aug	Cold/Warm	Initial / Query Size	Subset
Core-Set [34]	VGG16	RMSProp	0.001	**?**	cold	5,000 / 5,000	✗
VAAL [37]	VGG16	SGD	0.01	✓	?	5,000 / 2,500	✗
BADGE [4]	ResNet18/VGG11	Adam	0.001	✗	cold	100 / 100	✗
						100 / 1,000	✗
						100 / 10,000	✗
LLOSS [42]	ResNet18	SGD	0.1	✓	?	1,000 / 1,000	10,000
TA-VAAL [15]	ResNet18	SGD	0.1	✓	cold	1,000 / 1,000	10,000
CoreGCN [8]	ResNet18	SGD	0.1	✓	cold	1,000 / 1,000	10,000
ISAL [24]	ResNet18	SGD	0.1	✓	?	1,000 / 1,000	✗

Table 2: Experimental settings of state-of-the-art active learning.

A. Active Learning Methods

Selection strategies for active learning can be roughly categorized in (a) uncertainty-based sampling, (b) diversity-based sampling, and (c) combined approaches.

Uncertainty-based sampling. A classifier's uncertainty about a particular data point's prediction, directly relates to the lack of knowledge and thus the necessity for ground-truth. Uncertainty can, for instance, be measured by posterior probabilities [23, 39], the margin between the first and second most likely class [14, 32], or the entropy of the entire prediction [35]. By using the predicted output directly, uncertainty can also be estimated based on dropout neural networks [9] with Monte Carlo integration [27]. Yoo and Kweon [42], in turn, train an additional model to estimate the prediction's loss and most recently, Liu et al. [24] selects samples with the influence function which can approximate the change in model performance caused by newly added sample.

Diversity-based sampling. Additionally, sample diversity has been shown to be crucial for active learning as well [34, 37]. Core-Set [34] formulates active learning as coreset selection [1] to increasing diversity in a query batch, which can be proven optimal if the number of classes is small. By approximating the k-Center problem [6], Sener and Savarese [34] show that their approach is also effective in practice performing image classification. However, performance suffers for larger number of classes and high-dimensional data, which Sinha et al. [37] set out to address using variational autoencoders [17] in a conceptionally similar setting as generate adversarial networks (GAN) [10] to discriminate between labeled and unlabeled samples.

Combined approaches. Most recent research, however, attempts to strike a balance between sample diversity and uncertainty of the classifier. As an example, Kim et al. [15] propose to embed the predicted loss of a task learner [42] on the latent space of VAAL [37] via "Ranking Conditional GANs" [33]. BADGE [4] estimates the gradient regarding parameters in the final layer and uses k-means++ to simultaneously capture samples with high uncertainty (large gradients) and high diversity (more diverse gradient directions). The same authors have further extended this concept to a more general algorithm [3] via Fisher information matrices [38]. CoreGCN [8], in turn, exploits the characteristic of information sharing in Graph Convolution Network (GCN) [18].

B. Backbone Architecture

Table 3 summarizes the network definitions of the backbone architectures as used by recent active learning strategies. For most approaches, the exact architecture can be derived from the publication itself or from open-source implementations provided by the original authors. As mentioned in the main part, it is of utmost importance to carefully inspect and adapt implementations for a comparative evaluation. For BADGE, for instance, we observe that while the publication mentions the use of VGG11 the provided implementation [2] uses VGG16. Also, the architecture of the used ResNet18 differs from the other strategies and, thus, also from the original implementation as described by He et al. [11]. The number of filters in the convolutional layers is only a quarter of those in the implementations of TA-VAAL, LLOSS, and CoreGCN. In order to achieve reliable and comparable results, consistency of the backbone implementation is crucial.

C. Cold Start and Warm Start

Lang et al. [22] find that the difference between "cold starts" and "warm starts" quickly diminishes during training of AL methods. To verify this observation, we train all five AL methods on the CIFAR-10 dataset with both variants. Fig. 10 shows the results. With few training samples, the variance within a method is high and, thus, significantly differs between "cold starts" and "warm starts". As an example, at $3\,\mathrm{k}$ labeled samples, Entropy ranks first when trained with "warm starts", but barely improves over Random with "cold starts", ranking fourth. While there are clear differences in the early training stages, the difference between "cold"

Method	Type	Network Architecture
Core-Set [34]	VGGxx	VGG11, VGG13, VGG16, and VGG19 as proposed by Simonyan and Zisserman [36].
VAAL [37]	ResNet18	No details specified. No implementation available.
	VGG16	As proposed by Simonyan and Zisserman [36].
LLOSS [42] TA-VAAL [15] CoreGCN [8]	ResNet18	As proposed by He et al. [11]: conv 64 → conv 64 → conv 64 → conv 128 → conv 128 → conv 256 → conv 256 → conv 512 → conv 512 → avgpool
BADGE [4]	ResNet18	conv **16** → conv **16** → conv **16** → conv **32** → conv **32** → conv **64** → conv **64** → conv **128** → conv **128** → avgpool
	VGG11 (Paper)	No details specified.
	VGG16 (Code)	As proposed by Simonyan and Zisserman [36].

Table 3: Comparison of backbone architectures of state-of-the-art active learning strategies.

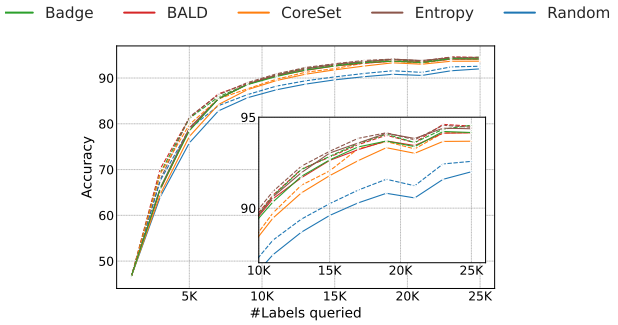


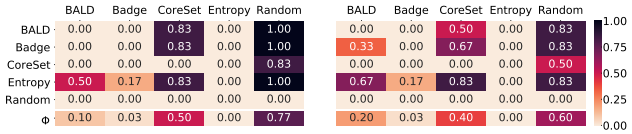
Figure 10: Comparison of "cold starts" (dashed lines) and "warm starts" (solid lines) for different AL methods.

and "warm starts" quickly fades away. With an increasing number of training samples, the accuracy curves of both types run almost parallel to each other. However, we find that "warm starts" have two major advantages over "cold starts". First, training requires less time and is computationally more effective. Second, "warm starts" stabilizes active learning as explained in Section 2.2.1 and, thus, allows to eliminate a crucial source of randomness.

D. Early Stopping

The number of training epochs brings randomness to the training process and can influence a comparative evaluation. Thus, a popular choice is to consider the number of training epochs as hyperparameter. As an example, Yoo and Kweon [42] have identified 200 epochs as a suitable configuration for learning ResNet18 on CIFAR-10, where the model is fully trained but does not overfit yet. As our experiments center over this exact configuration, we use this widespread setting [8, 15, 42].

In this section, we show the influence of using early stopping in comparison to a fixed number of epochs. Early stopping adjusts training to the necessities of the currently selected samples and thus is dependent on the AL method. While this procedure can be implemented in a various ways [26], we opt for a simple strategy that stops training if the current best validation accuracy does not increase for five epochs in a row. Fig. 11 shows the results of our evaluation for (a) early stopping and (b) a fixed number of 200 epochs using pair-wise penalty matrices.



(a) Early stopping (b) 200 epochs

Figure 11: Analysis of different training modes.

The performance of BALD, BADGE, and Entropy converge and become more similar to each other with early stopping than with 200 training epochs. Thus, the order can change in the sense that AL methods appear on a par. In Fig. 12, we inspect the number of training epochs when early stopping is triggered. None of the training cycles exceeds 200 epochs, stopping in the range of 162 to 198. Consequently, our fixed limit of 200 epochs ensures that the model is fully trained for all considered methods and all active learning cycles. While overfitting may occur, the extent is limited

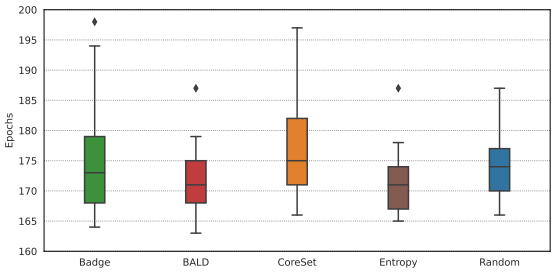
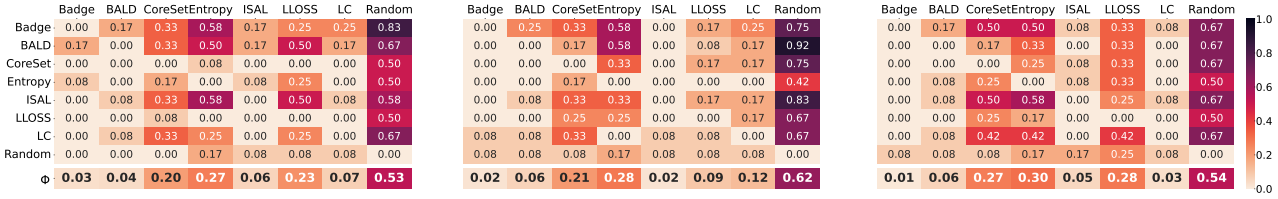
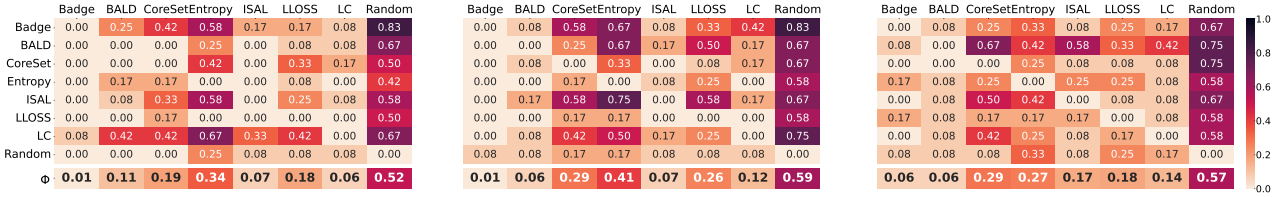


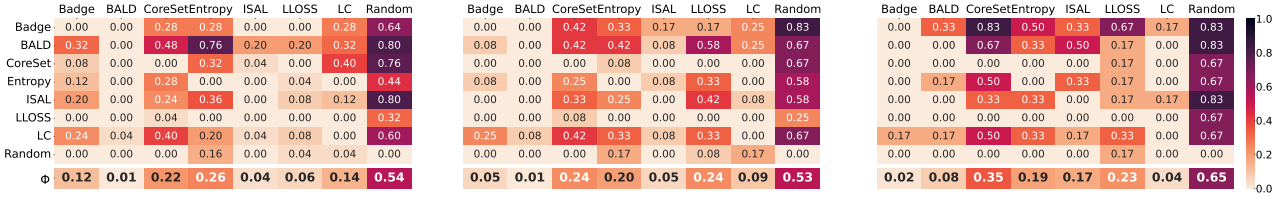
Figure 12: Training epochs of per cycle using early stopping.



(a) Init sets



(b) Model initialization



(c) Batch sizes

Figure 13: Analysis of active learning performance on CIFAR-100.

and the benefit of a leveled playing-field prevails. Early stopping can speed up the active learning process on the expense of a more variable experimental setup. We thus recommend selecting a fixed number of training epochs when conducting active learning evaluations, suitable for the used architecture and dataset, to ensure all methods yield a fully-trained methods irrespective of the early-stopping parameterization.

E. Model/Method Initialization (CIFAR-100)

As described in the main part, we also analyze the influence of initialization sets and model initialization on CIFAR-100. Fig. 13a shows pair-wise penalty matrices of three different init sets on CIFAR-100. For the second initial seed, LLOSS ($\varnothing$ 0.09) sightly outperforms LC ($\varnothing$ 0.12), while using the first and the third seed LC outperform LLOSS distinctly. Similar observations are obtained for model initialization as presented in Fig. 13b. Hence, the conclusion drawn in Section 2.2.1 based on CIFAR-10 also holds for CIFAR-100: Init-set seeds and model initialization can change the ranking of AL methods.

F. Non-/Deterministic Computations

In Fig. 14, we show the accuracy progression for the experiments on deterministic and non-deterministic computations as presented in Section 2.2.2. For non-deterministic training, the curves for Entropy and BADGE are very close, while for deterministic training, the two curves start off similar but diverge from 7 k labels on: Entropy begins to outperform BADGE and ends as the strategy with the best performance overall. Interestingly, for BALD and ISAL there hardly is a difference between deterministic and non-deterministic training.

G. Query-Batch Size (CIFAR-100)

We additionally run experiments on CIFAR-100 with query-batch sizes 1,000, 2,000, and 4,000. As shown in Fig. 13c, for 1,000 samples, LLOSS ($\varnothing$ 0.06) sightly outperforms LC ($\varnothing$ 0.14), while when using 2,000 and 4,000 samples, LC outperform LLOSS. Moreover, the performance of ISAL fluctuates significantly for different query-batch sizes. Our experiments on CIFAR-100 further demonstrate that the ranking of AL strategies varies with the query-batch size.

H. Subset Sampling (CIFAR-100)

Moreover, we investigate the effect of sub-sampling on CIFAR-100 and find that also here the ranking of AL strategies changes. In Fig. 15, we see that the performance of LLOSS and ISAL fluctuates significantly, but for LC, Entropy, and BADGE the results are better with sub-sampling than without. Note, that on CIFAR-10, we observe the opposite. We suspect that sub-sampling adds diversity at each round's selection process for datasets with large number of classes. For datasets with small number of classes, such as CIFAR-10, in turn, the limited size of selectable samples outweighs the gained diversity.

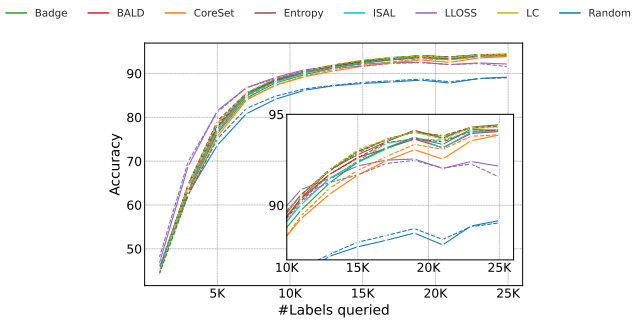
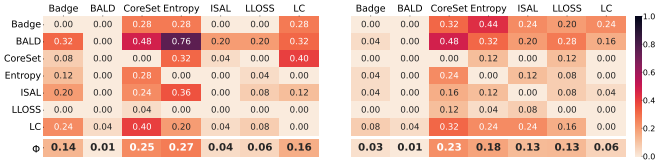
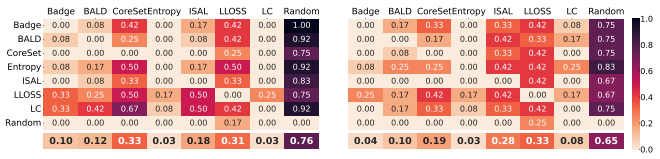


Figure 14: Comparison of non-deterministic (dashed lines) and deterministic computations for different AL methods.



(a) Full (b) Sub-sampled

Figure 15: Effect of sub-sampling on CIFAR-100.



(a) Ratio 10 (b) Ratio 100

Figure 16: CIFAR-10 with different imbalance ratios.

I. Imbalanced Datasets

For investigating the effect of imbalanced data on active learning strategies, we construct an imbalanced dataset based on CIFAR-10 and conduct experiments, using the same experimental settings as described in Section 2.

Dataset construction. We construct the imbalanced datasets as proposed by Kim et al. [15] and split CIFAR-10 into two halves, A and B, containing five classes each. One half (A) is kept as is, while we randomly sample inputs from the other half (B) up until we reach the predefined imbalance ratio, which is defined as the ratio of the first half and the sub-sampled "half", $^{A}/_{B'}$. This way, we construct two imbalanced datasets with a ratio of 10 and 100, denoted as iCIFAR-10_{10} and iCIFAR-10_{100}, respectively.

Evaluation. We use the balanced accuracy (BACC) [7] to more reliably measure performance on the imbalanced datasets. Based on these results, we then generate pair-wise penalty matrices for analyzing the different active learning methods, that are depicted in Fig. 16 for (a) iCIFAR-10_{10} and (b) iCIFAR-10_{100}. As also corroborated by Munjal et al. [25], AL methods show a varying degree of change in different imbalance settings, which is also noticeable in the statistic analysis. As an example, the performance of CoreSet and ISAL fluctuates significantly for the two different imbalanced ratios. For balanced CIFAR-10, LC ($\varnothing$ 0.01) outperforms Entropy ($\varnothing$ 0.04) while the ranking order converts on iCIFAR-10_{100}.

J. Scalability to TinyImageNet

Next, we study the performance of active learning on a large-scale dataset. TinyImageNet is a subset of ImageNet, containing 200 classes with 500 images per class. In contrast to our experiments on CIFAR-10 and CIFAR-100, here we use ResNext50 [41] as a backend model. We conclude that AL strategies have different scalability to TinyImageNet.

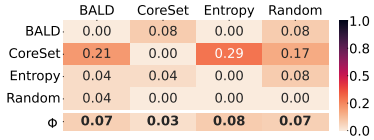


Figure 17: PPM on TinyImageNet.

We again leave BADGE out and report results for the remaining approaches as pair-wise penalty matrix in Fig. 17. Interestingly, Core-Set ranks best ($\varnothing$ 0.03) while BALD and Entropy fall behind jointly ($\varnothing$ 0.073), although for small-scale datasets its has been the other way around. We speculate that this is also attributed to the difference in uncertainty-based and diversity-based methods.

K. Social impact

The perhaps largest and most controversial aspect of active learning is its ecological footprint: We are trading labeling costs for computation, that is, we invest more computational resources to get along with as little labeled data as possible. While this makes sense from a pragmatic point of view, active learning causes a significant number of additional computations. Moreover, our work is particularly resource-intensive as we are evaluating multiple different aspects and settings of an already resource-intensive concept, enlarging the footprint even further.

We have conducted our experiments on NVIDIA A-100 and NVIDIA RTX-3090 GPU cards and have consumed about 3,900 GPU hours in total. This amounts to an estimated total CO_2 emissions of 594.75 kg CO_2eq when using Google Cloud Platform in region `europe-west3`. However, our university consumes 100 % renewable-energy, such that our specific CO_2 emissions for the project is 0.52 kg CO_2eq only. Estimates are conducted using the "Machine Learning Impact Calculator" [21].