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) VGG16 (Code)	No details specified. As proposed by Simonyan and Zisserman [36].

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

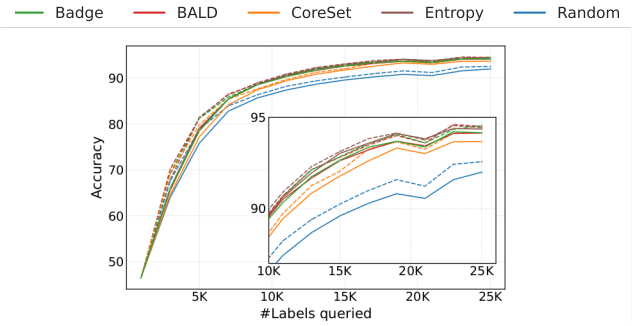


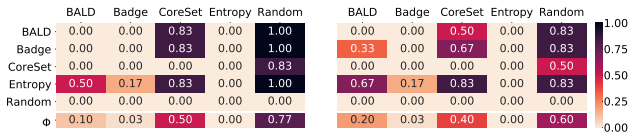
Figure 10: Comparison of "cold starts" (dashed lines) and "warm starts" (solid lines) for different AL methods.

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

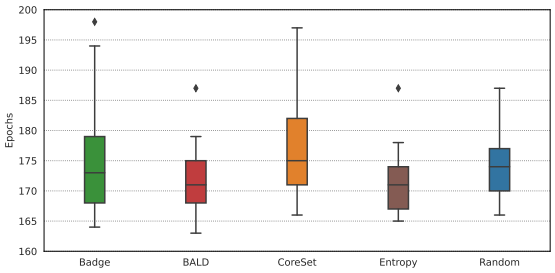
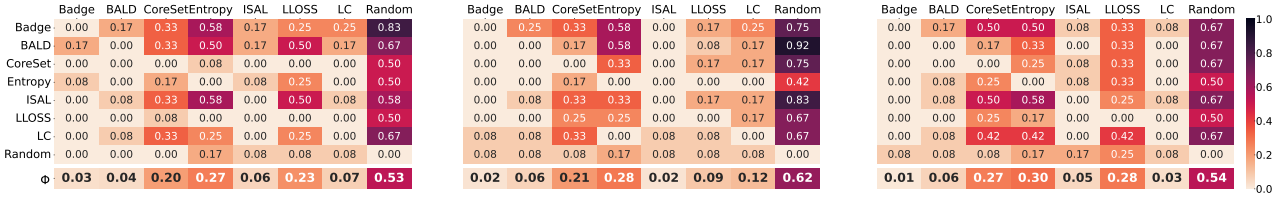
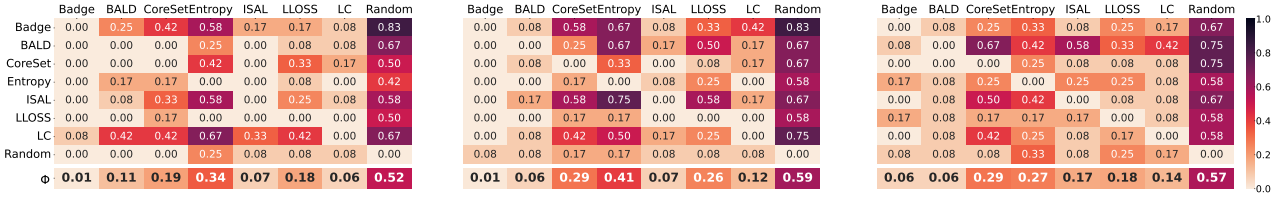


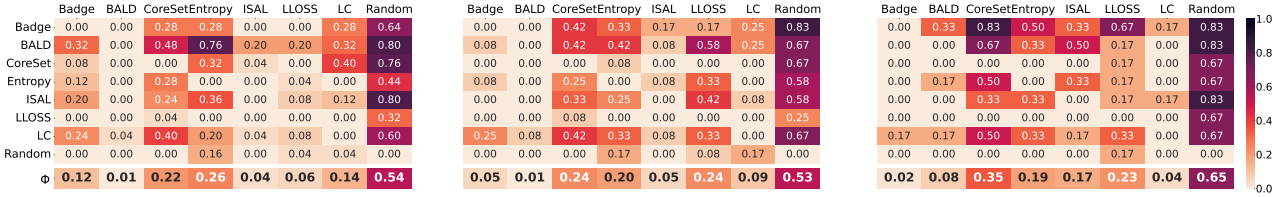
Figure 12: Training epochs of per cycle using early stopping.



(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.

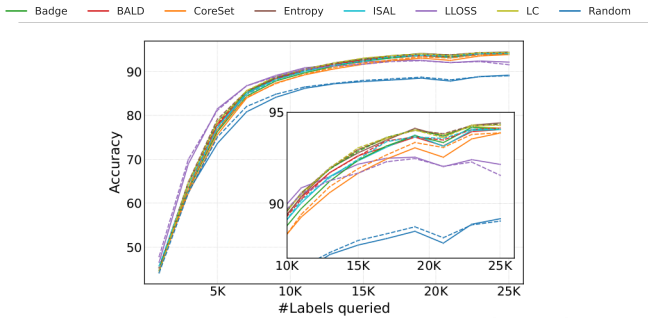
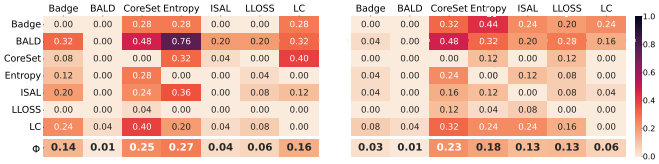
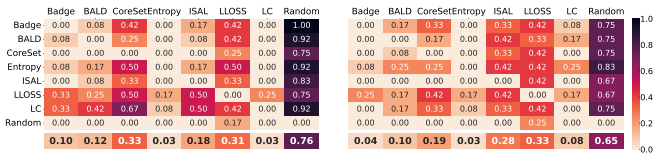


Figure 14: Comparison of non-deterministic (dashed lines) and deterministic computations for different AL methods.



(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.

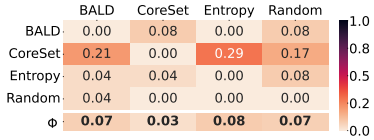


Figure 17: PPM on TinyImageNet.

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].