

---

# Effective Evaluation of Deep Active Learning on Image Classification Tasks

---

**Nathan Beck<sup>1</sup> Durga Sivasubramanian<sup>2</sup> Apurva Dani<sup>3</sup>**  
**Ganesh Ramakrishnan<sup>2</sup> Rishabh Iyer<sup>1</sup>**

<sup>1</sup>The University of Texas at Dallas <sup>2</sup>Indian Institute of Technology, Bombay

<sup>3</sup>AIFY Innovation Labs

{nathan.beck, rishabh.iyer}@utdallas.edu {durgas, ganesh}@cse.iitb.ac.in  
apurvadani@gmail.com

## Abstract

With the goal of making deep learning more label-efficient, a growing number of papers have been studying active learning (AL) for deep models. However, there are a number of issues in the prevalent experimental settings, mainly stemming from a lack of unified implementation and benchmarking. Issues in the current literature include sometimes contradictory observations on the performance of different AL algorithms, unintended exclusion of important generalization approaches such as data augmentation and SGD for optimization, a lack of study of evaluation facets like the labeling efficiency of AL, and little or no clarity on the scenarios in which AL outperforms random sampling (RS). In this work, we present a unified re-implementation of state-of-the-art AL algorithms in the context of image classification via our new open-source AL toolkit DISTIL<sup>1</sup>, and we carefully study these issues as facets of effective evaluation. On the positive side, we show that AL techniques are  $2\times$  to  $4\times$  more label-efficient compared to RS with the use of data augmentation. Surprisingly, when data augmentation is included, there is no longer a consistent gain in using BADGE, a state-of-the-art approach, over simple uncertainty sampling. We then do a careful analysis of how existing approaches perform with varying amounts of redundancy and number of examples per class. Finally, we provide several insights for AL practitioners to consider in future work, such as the effect of the AL batch size, the effect of initialization, the importance of retraining the model at every round, and other insights.

## 1 Introduction

Much of deep learning owes its success to the vast quantities of data used in training deep neural networks. Indeed, data sets used in training deep neural networks range from tens of thousands of data instances to millions of data instances. While using these large data sets continues to offer substantive performance benefits, the act of procuring labels for these data sets can be expensive and time-consuming depending on the task at hand. For example, medical imaging tasks for predicting cancer require the specialized skill-set of a radiologist, whose time is both hard to get and very expensive. Active learning (AL) tries to address this problem by sampling the smallest number of labeled examples needed to reach a desired test accuracy. In AL, the unlabeled data instances are intelligently chosen by a selection algorithm for a human annotator to label and present to an existing model. This selection is performed by using measures like uncertainty, diversity, or a combination.

Despite the amazing progress by various papers in deep AL, a number of issues remain. Firstly, a number of contradictory findings have been recorded in past work [21], including large differences in

---

<sup>1</sup><https://github.com/decile-team/distil>

accuracies of certain baselines and contradictory conclusions on the relative performance of certain algorithms (*e.g.*, coresnet approach [26] compared to random and uncertainty sampling). A lot of this is because different implementation details are often used (*e.g.*, differences in hyper-parameters, learning algorithms, *etc.*). Additionally, different papers have used different sets of baselines, and a few important baselines (*e.g.*, [30]) have rarely been used in these comparisons. Furthermore, most AL publications do not use data augmentation [5, 6, 24, 34], which significantly improves the generalization performance of deep models. The lack of data augmentation in most AL approaches such as those in [4, 26, 29] is often paired with the use of adaptive optimizers such as Adam [17] and RMSProp, which have been shown to hamper generalization performance [35, 31] in deep learning. Moreover, a number of important details and guiding principles for practitioners are not clear: **a)** Should model training be restarted after every round, or can one fine-tune the current model? **b)** What is the effect of using carefully crafted seed sets in AL? **c)** Is there an ideal AL batch size, or does the batch size actually matter? **d)** When is AL more effective than random sampling, and what are the factors that affect the labeling efficiency of AL algorithms? Lastly, **e)** how scalable are AL algorithms; specifically, what is the amount of time taken by model training versus that taken by AL selection, and can this be made more compute and energy-efficient? We study all these questions in the context of image classification tasks. To address the issues pointed out above, we provide a unified and modular re-implementation of most AL algorithms, building upon [11, 2, 23]. Below are some of the key findings and takeaways:

1. Data augmentation and other generalization approaches have a considerable impact on the test performance as well as on the labeling efficiency. For a ResNet-18 [9] model applied on the CIFAR-10 [18] data set, we show that most AL approaches with data augmentation achieve  $> 90\%$  accuracy with around 13k labeled points for CIFAR-10 while achieving labeling efficiencies of  $2\times$  to  $4\times$  on most data sets. *Data augmentation not only improves the test performance significantly but also improves the labeling efficiency with respect to RS.*
2. We observe that *SGD performs and generalizes better than Adam [17] consistently in AL* (which corroborates past work [35, 31]).
3. In the presence of data augmentation and the SGD optimizer, there is no significant advantage of diversity over very simple uncertainty sampling (at least in most standard academic data sets). *Specifically, the state-of-the-art approach BADGE [4] does not consistently outperform uncertainty sampling approaches such as entropy [27] and least confidence [27].*
4. When we make the data set artificially redundant (either by repeating data points or by using near repetitions through augmentations), we see that BADGE starts outperforming uncertainty sampling. *The difference between the two increases as the amount of redundancy increases, which suggests that for data sets with a lot of redundancy (*e.g.*, frames from a video), it makes sense to use an approach such as BADGE.*
5. We see that the number of instances per class has an important impact on the performance of AL algorithms: *The fewer the examples there are per class, the smaller the room there is for AL to improve over random sampling*, which further leads to reduced labeling efficiency.
6. We study the role of labeled set initialization and the choice of the AL batch size. We find that the initialization of the labeled seed set (*e.g.*, a random versus a more diverse or representative seed set) has little to no impact on the performance of AL after a few rounds; likewise, reasonably sized choices of the AL batch size also have little to no impact.
7. We observe that updating the models from previous rounds (fine-tuning) versus retraining the models from scratch negatively impacts the performance of AL *only in the early selection rounds* – the performance difference between the two strategies vanishes once the model has stabilized.
8. Finally, we study the turnaround time and the energy consumption of AL. We find that the most time-consuming and energy-inefficient part of the AL loop is the model (re)training. We suggest possible strategies to accelerate this component of AL by applying our findings on the impact of AL batch size and model fine-tuning, and we explore the use of data subset selection (DSS) techniques [15] for accelerating model training. We show that these improve the training time by  $3\times$  with negligible accuracy loss.

Hence, we contribute a series of observations on our chosen facets of evaluation that can help address shortcomings in effective evaluation of active learning. Via our contribution, we hope to encourage future practitioners to evaluate AL using our observations on the evaluation facets listed in Sec. 2.

## 2 Facets of Effective Evaluation

To set the context for our key findings, we enumerate some facets of effective evaluation of AL.

**P1: Unified Experimental Setting.** An effective comparison of AL selection algorithms requires attention to the experimental setting. Since several experimental parameters such as the choice of optimizer, the learning rate and its schedule, AL batch sizes, labeled set initializations, and other parameters have varied across publications, it is important to present a unified experimental setting wherein many of the re-implemented AL approaches can be compared. Luckily, many AL approaches can be decoupled from the training loop since these methods only consider the batch query. In this fashion, we are able to control all aspects of intermediate training and common parameters used in AL batch selection. In this work, we utilize **DISTIL**, our unified re-implementation of most state-of-the-art AL approaches, in a number of experimental settings. We build upon the Deep Active Learning GitHub repository [11, 2], add several recent algorithms [30, 16, 29], and make the code modular to enable the use of different training loops. We provide the code along with Jupyter notebooks to reproduce the experiments from this paper in Appendix A. Detailed results are in **Sec. 4.1**.

**P2: Emphasis on Labeling Efficiency.** Most AL works present their results by comparing the test accuracy with respect to the number of labeled points. While this perspective highlights the benefit of a selection algorithm compared to their baselines, it does not explicitly show the labeling efficiency of that algorithm with respect to random sampling. Since AL is premised on obtaining a target test accuracy using the fewest labels possible, it must be clear how many fewer labels a selection algorithm needs in order to achieve that target test accuracy with respect to random sampling. In this paper (specifically in **Sec. 4.1** and **Sec. 4.2**), we emphasize this measure through the use of labeling efficiency plots and motivate the need for studying labeling efficiency in future works as a means to more holistically understand AL selection algorithms. We give a concise definition of labeling efficiency in **Sec. 4.1**.

**P3: AL with Data Augmentation.** Next, we study the effect of data augmentation [5, 6, 24, 34]. Data augmentation has become commonplace in deep learning but has hardly been studied or used in AL algorithms; most AL papers do not report results with data augmentation. For example, the recent state-of-the-art BADGE [4] approach achieved only slightly more than 80% towards the end of AL whereas state-of-the-art training methods (with data augmentation) achieve more than 90% for most model families. We show that data augmentation approaches such as random cropping, random horizontal flips, and other transformations when used in AL training loops help achieve 90% accuracy with only 13k labeled points in CIFAR-10 (26% of the data set). *Data augmentation not only increases the accuracies achieved using AL but also improves the labeling efficiency. Somewhat surprisingly, we also see that BADGE-like diversity-based approaches no longer outperform simple entropy sampling [27]*. We hypothesize that the augmentations help provide diversity to the selected data instances in the low-redundancy settings offered by the data sets commonly used in academia. We report these experiments in **Sec. 4.2**.

**P4: AL with SGD and Other Generalization Approaches.** Another facet of effective evaluation of AL involves analyzing the effect of the optimizer. Notably, a common observation is that Adam [17] generalizes poorly compared to SGD [31, 35], but we also find that most AL approaches [4, 26, 29] have used adaptive optimizers like Adam and RMSProp. Knowing this fact and that data augmentation helps the generalization performance of AL, we ask if choosing *SGD over Adam [17] also improves the generalization performance of AL*. We explore how SGD compares against Adam and find that *Adam often results in reduced labeling efficiency and generalization performance despite being faster to converge than SGD (Sec. 4.3)*. To complement our observations concerning the generalization benefit of SGD, we also explore the effect of stochastic weight averaging [13] and shake-shake regularization [8] in AL, noting that both techniques further increase generalization performance. We report this in Appendix C.

**P5: Redundancy.** As a result of the growth of deep learning data sets, redundant data has become a more prevalent phenomenon in deep learning. However, many of the experiments performed to assess the quality of AL algorithms neglect accounting for redundant data sets. As a result, the reported performance of these algorithms might not be indicative of their actual performance in real-world settings. We carefully examine the effect of redundancy in AL algorithms in **Sec. 4.4** by progressively duplicating points in the CIFAR-10 data set [18]. We also consider another similar redundancy setting in Appendix D where we instead add augmented copies of data instances with subtle random translations and random cropping. We perform all experiments with *data augmentation* in the training. *We note that diversity-based methods such as BADGE [4] are robust against redundant data while uncertainty-based methods such as entropy sampling [27] poorly handle redundant data.*

**P6: When does AL offer benefit compared to random sampling?** Most past works in AL have not provided guidelines as to when AL yields benefit over random sampling. In this paper, we

hypothesize that the number of instances per class in the unlabeled data set provides a cue as to how much benefit AL can have. We are motivated by a simple observation (as we show in Sec. 4.1) that AL achieves a labeling efficiency of  $1.8\times$  to  $2\times$  on CIFAR-10 [18] but only  $1.3\times$  on CIFAR-100 [18]. To provide a clear experiment to test this hypothesis, we explore a scenario in Sec. 4.5 where we restrict the number of examples in CIFAR-10 [18] on a per-class basis in the unlabeled set. We study the effect that this restriction has on the labeling efficiency of AL, *noting that having fewer instances per class results in reduced labeling efficiency.*

**P7: AL Batch Size and Seed Set Initialization.** Across AL publications, different choices for the AL batch size are used. To draw an effective evaluation of AL, the effect of the AL batch size needs to be studied. Namely, is there a difference in the accuracies obtained by AL in practical scenarios if the AL batch size is varied? Furthermore, the labeled seed sets used in many AL experiments are often random; however, other sophisticated methods to construct higher-quality labeled seed sets (*e.g.*, representative seed sets) can be used. In this work, we examine the effect of the AL batch size and the labeled seed set on the accuracies obtained by AL. These experiments are conducted in Sec. 4.6; the high-level takeaway is that *neither reasonable choices of the AL batch size nor the initial seed set play a significant role in the performance (accuracies and labeling efficiencies) of AL.*

**P8: Model Updating versus Model Retraining.** Another facet of effective evaluation of AL involves analyzing the effect of updating the model between AL rounds versus retraining the model from a random initialization. Previous work done in [3] suggests that warm-starting in deep learning can hurt the generalization performance of these models; however, their study was performed without data augmentation and with the Adam optimizer. We explore the effect of model updating between selection rounds while using SGD and data augmentation. *We note that the test performance is negatively affected for the early rounds of AL when doing model updating versus model retraining; however, this negative effect diminishes once the model has matured and stabilized.* This experiment is conducted in Sec. 4.7. Furthermore, we observe that when we use Adam and no data augmentation (Appendix E), we see a larger and more consistent degradation of the test performance of model updating versus model retraining, which confirms the observation of [3].

**P9: Scalability.** AL attempts to solve the labeling problem in large data sets, and large-scale usages of AL require that both the selection algorithm and the training procedure scale to the needs of these large data sets. Hence, we study the time spent in computing the query selection and in performing the intermediate model training, *observing that the dominating time factor of most AL configurations is often the intermediate model training.* Accordingly, scalable AL usages can then draw upon our previous observations concerning AL batch size and model updating. Namely, utilizing larger AL batch sizes whenever possible can reduce the number of intermediate training rounds needed, and opting for model updating can reduce the amount of time spent in intermediate model training. To alleviate the possible performance degradation that accompanies model updating, we further study the use of data subset selection (DSS) techniques to accelerate training. Specifically, we utilize GRAD-MATCH [15] in most intermediate training rounds, occasionally performing full training in a few intermediate rounds to evaluate the performance of each selection algorithm. We show in Sec. 4.8 that, on top of achieving significant speedups, *there is little to no performance degradation by using some of the above approaches with AL.*

### 3 Deep Active Learning Algorithms

Similar to supervised learning, AL starts with a labeled seed set that is used to train an initial model. Unlike supervised learning, there is another set of unlabeled instances  $\mathcal{D}_{UL}$  from which AL periodically selects instances to label and add to the existing labeled set. The AL algorithm uses information from the model, from the unlabeled set, or from both when selecting these instances to yield the largest increase in test performance after the model is updated or retrained. To better understand the main approaches compared in this paper, we introduce some notation that we will use in our description of each algorithm. Let  $f$  denote the deep neural network used in training,  $f_l$  denote the output of  $f$  at layer  $l$ , and  $\theta_l$  denote the parameters of layer  $l$ . We use  $b$  to denote the AL batch size and  $L$  to denote the number of layers of  $f$ .

**Uncertainty sampling** selects instances from  $\mathcal{D}_{UL}$  that maximize a measure of uncertainty on the model’s prediction of the instance’s label. Common uncertainty-based sampling techniques are **entropy**, **least confidence**, and **margin sampling** techniques [27]. Using  $p(x) = \text{softmax}(f_L(x))$ , each computes the following for each  $x \in \mathcal{D}_{UL}$ :

$$\begin{aligned}
\textbf{Entropy:} \quad H(x) &= -\sum_i p(x)_i \log p(x)_i \\
\textbf{Least Confidence:} \quad C(x) &= \max_i p(x)_i \\
\textbf{Margin:} \quad M(x) &= p(x)_{\sigma_1} - p(x)_{\sigma_2}
\end{aligned}$$

where  $\sigma_1$  and  $\sigma_2$  denote the indices of the largest and second-largest components of  $p(x)$ , respectively. **Entropy sampling** chooses  $x \in \mathcal{D}_{UL}$  associated with the  $b$  largest  $H(x)$  values. **Least confidence sampling** and **margin sampling** choose  $x \in \mathcal{D}_{UL}$  associated with the  $b$  smallest  $C(x)$  and  $M(x)$  values, respectively. By selecting uncertain points, future training on  $f$  can further refine the decision boundary, giving improved performance.

Next, the **CORESET** approach tries to find instances that represent or capture the structure of  $\mathcal{D}_{UL}$  by minimizing a core-set loss [26]. The authors of [26] show that this can be achieved by solving the following:

$$\min_{\mathbf{s}^1: |\mathbf{s}^1| \leq b} \max_i \min_{j \in \mathbf{s}^1 \cup \mathbf{s}^0} \Delta(x_i, x_j) \quad (1)$$

where  $\Delta$  denotes some distance metric,  $\mathbf{s}^1$  denotes the chosen AL batch, and  $\mathbf{s}^0$  denotes the labeled set. The authors of [26] use a robust  $k$ -center algorithm to solve this objective. The points selected in this fashion are thus diverse and representative of  $\mathcal{D}_{UL}$ , allowing for future training to produce  $f$  that generalizes better on the domain. In our experiments, we choose to represent each  $x$  by its penultimate layer embedding  $f_{L-1}(x)$  when calculating  $\Delta(x_i, x_j)$ .

**FASS** [30] applies a two-step process of uncertainty sampling and submodular selection. Specifically, the top  $b \times k$  most uncertain  $x \in \mathcal{D}_{UL}$  are initially chosen and stored in  $\mathcal{U}$ , where  $k$  is a configurable hyper-parameter. Next, a submodular [12] set function  $r : 2^{\mathcal{U}} \rightarrow \mathbb{R}$  is instantiated on  $\mathcal{U}$ . Submodular function maximization subject to cardinality constraints is known to find diverse or representative subsets; thus, **FASS** [30] selects  $\mathbf{s}^1 \subseteq \mathcal{U}$  such that  $|\mathbf{s}^1| = b$  using submodular maximization with cardinality constraints on  $r$ , thereby capturing both uncertainty (via  $\mathcal{U}$ ) and diversity (via  $r$ ) in  $\mathcal{D}_{UL}$ . In our experiments, we use **entropy sampling** [27] for the uncertainty measure and **facility location** [12] for  $r$ . As before, we use penultimate layer embeddings given by  $f_{L-1}(x)$  for each  $x \in \mathcal{U}$  when instantiating  $r$ .

**BADGE** [4], a very recent approach free of hyper-parameter tuning, first computes the last linear layer gradients of the hypothesized loss on each of the instances in  $\mathcal{D}_{UL}$ . Specifically, if we denote  $\hat{y}(x) = \max_i f_L(x)_i$  and  $\mathcal{L}$  as the loss used, then for each  $x \in \mathcal{D}_{UL}$ , we compute  $g(x) = \nabla_{\theta_{L-1}} \mathcal{L}(\hat{y}(x), f_L(x))$ . Denoting  $\mathcal{G} = \{g(x) | x \in \mathcal{D}_{UL}\}$ , **BADGE** [4] then runs the K-MEANS++ initialization algorithm [1] on  $\mathcal{G}$  to obtain  $b$  centers, which are likely to all have large loss gradient magnitudes (which captures uncertainty brought about by large changes to the loss) and diverse loss gradient directions (which captures diversity brought about by diverse parameter updates). The points corresponding to these  $b$  centers are then chosen as the AL batch by **BADGE** [4], which helps future training on  $f$  better adapt to the domain.

Lastly, **GLISTER-ACTIVE** [16] selects points by solving the following bi-level optimization problem:

$$\operatorname{argmax}_{\mathbf{s}^1 \subseteq \mathcal{D}_{UL}, |\mathbf{s}^1| \leq b} LL(\operatorname{argmax}_{\theta} LL(\theta, P(\mathbf{s}^1)), \mathcal{V}) \quad (2)$$

where  $LL(\theta, S)$  denotes the log-likelihood on the set  $S$  using  $f$  parameterized by  $\theta$ ,  $P(S) = \{(x, \hat{y}(x)) | x \in S\}$ , and  $\mathcal{V}$  denotes a validation set of interest. In our experiments, we set  $\mathcal{V}$  as the union of the labeled set and  $\mathcal{D}_{UL}$  with hypothesized labels (e.g., the set  $P(\mathcal{D}_{UL})$ ). Here, the inner optimization problem learns model parameters on the pseudo-labeled AL batch, and the outer optimization problem selects a set of unlabeled instances from  $\mathcal{D}_{UL}$  that maximizes the log-likelihood with respect to the validation set using the trained model parameters from the inner problem. The authors of [16] use an online meta approximation algorithm to carry out the bi-level optimization. Ultimately, **GLISTER-ACTIVE** [16] selects  $\mathbf{s}^1$  that is representative of the domain (since we use our specific choice of  $\mathcal{V}$ ), which aids in producing  $f$  that generalizes better on the domain.

Since the aim of deep AL methods is to reduce the labeling cost, AL is therefore most useful when considered in large data settings. Techniques such as BALD [10], adversarial-based AL techniques [7], and VAAL [29] are not as scalable; therefore, we do not consider them in most of our experiments in our main sections. For larger AL batch sizes, BALD requires intractably large collections of Monte Carlo samples for its Bayesian estimates, while VAAL [29] requires maintaining

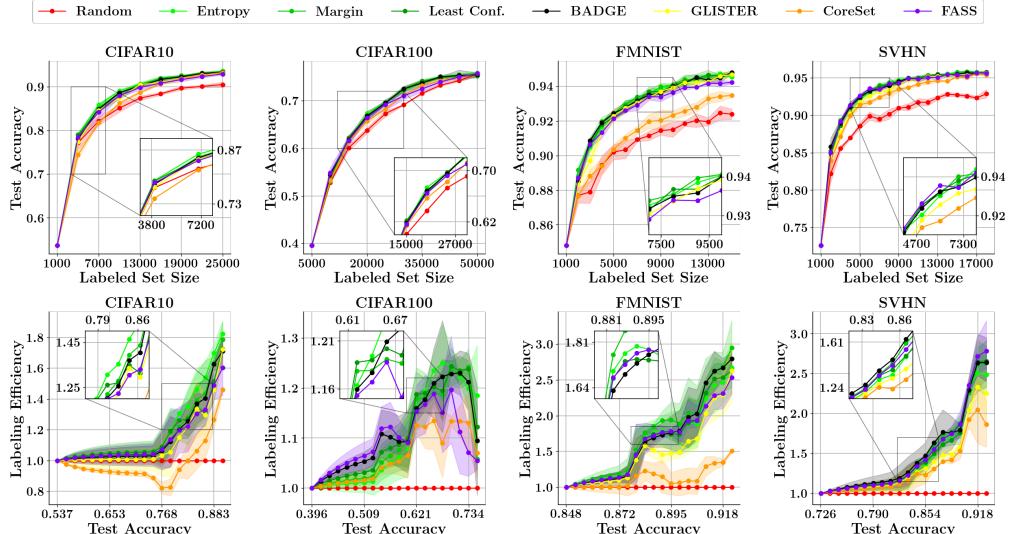


Figure 1: Baseline results on the CIFAR-10, CIFAR-100, Fashion-MNIST, and SVHN. We observe that most AL schemes achieve labeling efficiencies ranging from  $1.3\times$  to  $3\times$  compared to RS.

and training three models – the main supervised model, the VAE model, and the discriminator [29]. Additionally, other adversarial methods such as those studied in [7] require expensive perturbations to perform AL selection. For completeness, we perform a few small-scale experiments to compare all AL baselines (including BALD, VAAL, and other adversarial methods) in Appendix B. Specifically, we contrast the running times, test performance, and labeling efficiencies of all these AL approaches.

## 4 Experiments

In this section, we report experiments and results to study the previously listed facets of effective evaluation (**Sec. 2**) using **DISTIL** as discussed earlier in this paper. Furthermore, we use a common set of experimental parameters unless otherwise specified, which are detailed in Appendix A. Our data augmentations consist of random horizontal flips and random croppings ( $p = 0.5$ ) of the images padded by 4 pixels on each side, followed by data normalization. These augmentations occur every time a data instance is retrieved during training. To encourage reproduction of our results, we provide simple instructions to execute our experiments and list the compute resources needed to run these experiments in Appendix A. We evaluate our algorithms on the following five image classification data sets: CIFAR-10 [18], CIFAR-100 [18], MNIST [19], Fashion MNIST [32], and SVHN [22]. For each experiment, we initialize each AL strategy with an identical initial model and initial labeled seed set. Each model is trained until either the training accuracy reaches 99%, the epoch count reaches 300, or the training accuracy does not improve after 10 epochs. On all data sets except MNIST [19], a ResNet-18 [9] and a VGG-11 [28] are used. On MNIST [19], we use a model similar to a LeNet [19] (see Appendix A). Lastly, we run each experiment three times and report the average across all runs and the associated variance. We allude to significance test results here and provide details of the same in Appendix G.

**4.1 Baseline Experiments & Labeling Efficiency.** To have an accurate comparison of AL selection algorithms, we first conducted a baseline experiment across multiple data sets to gauge the effectiveness of each algorithm in a general scenario. As highlighted in **Sec. 3**, we restrict our comparison to random sampling, entropy sampling, margin sampling, least confidence sampling, BADGE, GLISTER-ACTIVE, CORESET, and FASS. Other algorithms are compared in Appendix B. In Figure 1 (upper plots), we plot the obtained test accuracy as a function of the size of the labeled set. In addition, in the lower plots, the labeling efficiency of each selection algorithm with respect to random sampling is displayed. Specifically, the labeling efficiency for a given test accuracy is computed as the ratio of the minimum number of labeled training instances needed for an AL algorithm to reach that accuracy to the minimum number of labeled training instances needed by random sampling to reach that accuracy. The results in Figure 1 are obtained using a ResNet-18 model on CIFAR-10, CIFAR-100, SVHN, and Fashion-MNIST. The results for VGG-11’s performance and the results for the MNIST data set are in Appendix B. In each baseline, *we observe that the top-*

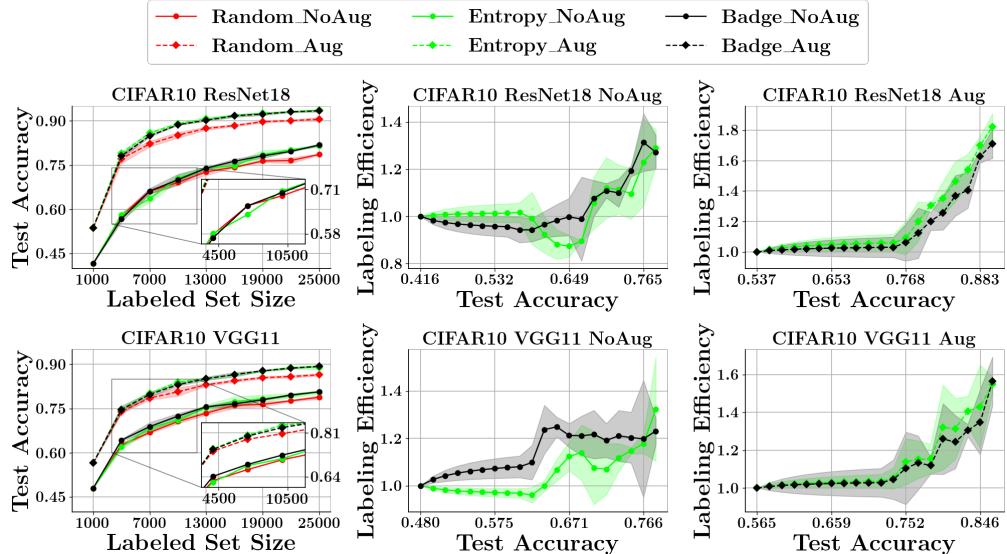


Figure 2: Comparing AL performance of ResNet-18 (top) and VGG-11 (bottom) on CIFAR-10 with and without augmentation. Data augmentation not only increases test accuracy but also improves the labeling efficiencies of AL. Furthermore, BADGE outperforms entropy sampling without data augmentation, but BADGE loses its advantage over entropy sampling when data augmentation is used.

performing strategies exhibit labeling efficiencies that range from  $1.3\times$  (CIFAR-100) to  $3\times$  (SVHN and FMNIST), indicating that the commonly used AL selection algorithms achieve their intended purpose of reducing labeling costs. For MNIST (Appendix B), we observe a labeling efficiency of  $4.5\times$ . This addresses **P1** (unified experimental setting) and **P2** (labeling efficiency).

**4.2 Data Augmentation.** To better understand the effect of data augmentation on image data sets in an AL setting, we compare AL on CIFAR-10 [18] with and without the use of data augmentation. We do this for BADGE and entropy sampling, which are the best-performing strategies in Figure 1 and are also representative techniques among uncertainty and diversity-based approaches. The results are shown in Figure 2. First, the use of augmentation significantly and consistently improves the accuracy by almost 10% when compared to no augmentation. Secondly, augmentation also increases the labeling efficiency of AL. The labeling efficiency improves from  $1.4\times$  to  $1.8\times$  for ResNet-18 and  $1.3\times$  to  $1.7\times$  for VGG-11 towards the end. Thirdly, we note that while BADGE actually outperforms entropy sampling in the no data augmentation case (which corroborates the results shown in [4]), there is no significant difference between the two with the addition of data augmentation (entropy sampling performs better than BADGE in just one of the selection rounds with  $\alpha = 0.05$ ). These results can be partly explained by the fact that data augmentation naturally introduces more diversity into the training. This addresses **P3**.

**4.3 Optimizer and Other Generalization Techniques.** Another aspect of AL evaluation involves analyzing the effect of the training strategy used. Past work in AL has mostly used adaptive gradient methods for optimization, such as Adam [4, 29] and RMSProp [26]. Given our analysis on generalization performance, it is natural to then consider the effect of using SGD over Adam since a number of works have argued that deep neural networks generalize better when trained with SGD as opposed to Adam [31, 35]. We study this in the context of AL and with the use of data augmentation. In particular, we run our experiment on CIFAR-10 [18] with ResNet-18 and compare Adam against SGD. For SGD, we use a momentum of 0.9, a weight decay of 0.0005, and a cosine-annealing learning rate ( $T_{max} = 300$ ). For Adam, we use no weight decay and the typical choice of  $\beta_1, \beta_2$  as 0.9 and 0.999, respectively. For both, we use a learning rate of 0.01. Based on our results (Figure 3),

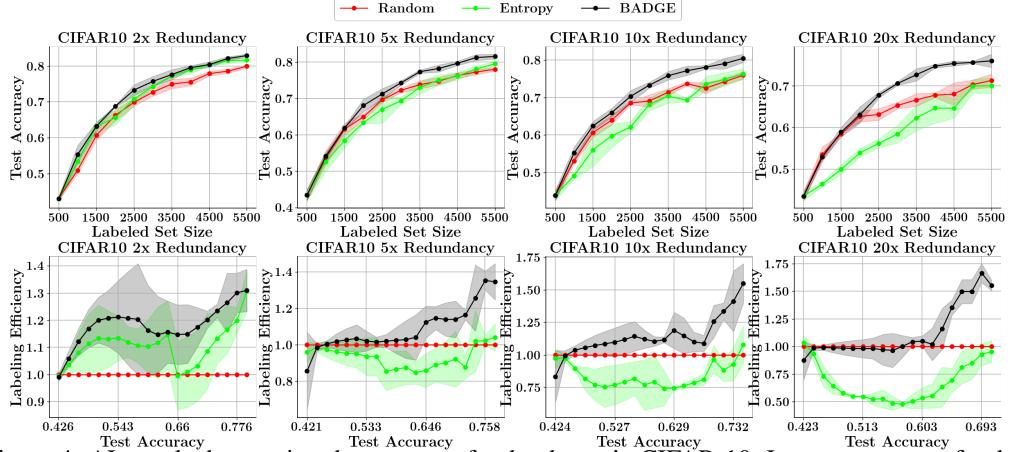


Figure 4: AL results by varying the amount of redundancy in CIFAR-10. Larger amounts of redundancy affect the performance of entropy sampling more than BADGE.

*we see that Adam results in reduced test accuracy and lower labeling efficiency compared to SGD; however, it converges faster than SGD* (for example, entropy sampling with SGD scores higher than with Adam in all rounds except the initial round with  $\alpha = 0.05$ ). Hence, the observations from [31, 35] seem to also apply in the AL setting. Finally, we study the effect of some more advanced generalization approaches, *viz.*, stochastic weight averaging (SWA) [13] and shake-shake regularization [8] in Appendix C, and we see that the takeaways are consistent: *These approaches all improve the test performance of AL while maintaining high labeling efficiency.* This addresses **P4**.

**4.4 Active Learning with Redundancy.** AL strategies are studied on academic data sets that are often not representative of large-scale data sets in real-world applications. Such data sets often contain many redundant data instances. To effectively evaluate these AL strategies, we rerun our baseline setting on a variant of CIFAR-10 [18] where a set of unique data instances is kept fixed while another disjoint set of data instances is duplicated until the original size of 50k training instances is reached. We construct this data set such that the original class ratios are not altered. Consequently, this allows us to evaluate the effect of redundancy under the scope of **P5**. In all our experiments, we keep 5k unique data instances fixed. Figure 4 shows the results for this setting when the duplicated set is repeated  $2\times, 5\times, 10\times$ , and  $20\times$ . *We see that entropy sampling [27] progressively does worse when more redundancy is introduced whereas BADGE [4] exhibits robustness to redundancy.* Furthermore, the more redundancy there is, the more the labeling efficiency of BADGE increases towards the end. We summarize by concluding our observation in **P5**: *Diversity-based methods are robust against redundant data while uncertainty-based methods poorly handle redundant data.* In Appendix D, we repeat this experiment by generating redundancies using augmentations on existing instances (random cropping and translation) as opposed to simple duplication – we observe the same takeaways as above.

**4.5 Effect of Examples Per Class.** Since the labeling efficiencies achieved in CIFAR-100 ( $1.3\times$ ) are much lower than those achieved in CIFAR-10 ( $1.8\times$ ), we hypothesize that the number of examples per class in the unlabeled set plays an important role in how well AL works compared to RS. We suspect that having more examples per class in the unlabeled set allows for AL to make better choices in refining class performance. To verify this hypothesis, we conduct an experiment on CIFAR-10 and Fashion-MNIST wherein we vary the number of unlabeled data instances per class from 1k up to 3k. Results for this experiment are presented in Figure 5. We see that *having a higher number of data instances per class results in higher relative test accuracies and labeling efficiencies with respect to random sampling.* Hence, scenarios in which the unlabeled set is suspected to have few instances per class may be susceptible to reduced AL benefit as stated in **P6** and as illustrated in this experiment.

**4.6 Effect of Varying AL Batch Sizes and Initializations.** To better understand how the choice of the AL batch size affects the evaluation of AL algorithms, we run an experiment on CIFAR-10 where the AL batch size is varied between 1000, 3000, and 6000 queried instances. The results are presented in Table 1. We see that *the AL batch size has little effect on each selection algorithm’s obtained test accuracies and labeling efficiencies* (no two configurations of the AL batch size had accuracies significantly different with  $\alpha = 0.05$  for more than 20% of the selection rounds). These

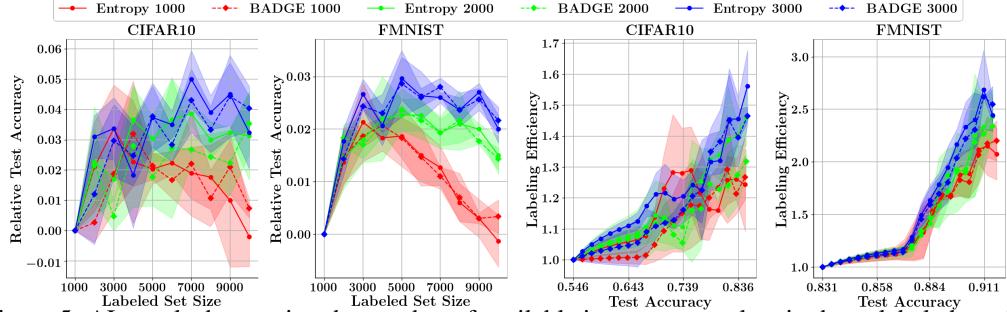


Figure 5: AL results by varying the number of available instances per class in the unlabeled portion of CIFAR-10. The left two plots show relative test accuracies (compared to RS) while the right two show labeling efficiencies. 1000, 2000, and 3000 in the legend correspond to the number of examples per class in the unlabeled set. The more the number of examples there are per class, the better the relative test accuracy & labeling efficiency of AL is compared to RS.

Alg_Budget	1000 points	7000 points	13000 points	19000 points	25000 points
RAND_1000	55.6 ± 0.0	82.5 ± 0.5	87.8 ± 0.6	89.1 ± 0.8	91.0 ± 0.2
RAND_3000	55.6 ± 0.0	82.2 ± 1.2	87.0 ± 0.2	89.9 ± 0.5	90.5 ± 0.5
RAND_6000	55.6 ± 0.0	82.0 ± 0.8	87.7 ± 0.7	89.4 ± 0.7	90.7 ± 0.6
ENT_1000	55.6 ± 0.0	84.3 ± 0.9	88.6 ± 1.2	91.9 ± 0.9	92.8 ± 0.8
ENT_3000	55.6 ± 0.0	83.3 ± 1.4	89.4 ± 1.2	91.7 ± 0.8	92.5 ± 0.8
ENT_6000	55.6 ± 0.0	82.7 ± 1.0	88.5 ± 0.1	91.0 ± 1.1	92.4 ± 0.8
BADGE_1000	55.6 ± 0.0	83.5 ± 1.5	89.0 ± 0.6	90.8 ± 1.2	92.4 ± 0.6
BADGE_3000	55.6 ± 0.0	83.4 ± 1.3	88.4 ± 0.9	91.5 ± 0.7	92.6 ± 0.5
BADGE_6000	55.6 ± 0.0	82.13 ± 1.4	88.1 ± 0.6	91.1 ± 0.9	92.3 ± 1.1

Table 1: Test accuracy and standard deviation on CIFAR-10 across varying AL batch sizes. We find that reasonable choices of the batch size make little difference in the achieved test accuracies.

results show that the choice of AL batch size is not heavily linked with an algorithm’s labeling efficiency. Thus, the choice of the AL batch size can be made to optimize the labeling-training pipeline in real-world settings.

Almost all AL strategies assume a warm start; *i.e.*, a small labeled seed set and a model trained on this set are already given. We next study the effect of choosing a sophisticated seed set on the generalization performance and labeling efficiency of AL. In this experiment, we compare a seed set formed by a facility location selection (to select a representative subset) [12, 14] against a randomly chosen seed set by repeating the baseline setting with these seed sets. The results are shown in Table 2. We observe that the carefully constructed seed sets offer a better initial accuracy; however, this benefit is quickly lost after only a few rounds of AL (entropy sampling with a random initialization versus entropy sampling with a facility location initialization is the only comparison to show a significant difference in test accuracy in a selection round with  $\alpha = 0.05$ ). *We conclude that the seed set initialization does play a role very early on, but its influence diminishes after a few rounds of AL as stated in P7.*

**4.7 Model Reset vs. Update.** We now analyze a strategy for reducing the training time wherein we update (fine-tune) the model from the previous rounds of AL. We perform this experiment on CIFAR-10, and we compare the update strategy with the resetting strategy. We see that the initial test accuracy is worse for every AL strategy when the model is updated in every round as opposed to when it is reset at every round. This is also corroborated by [3], wherein warm-starting deep models has the propensity to hurt generalization performance. *However, we interestingly observe that the resetting strategy as well as the updating strategy converge to similar test accuracies at later rounds, implying that it does not matter as much whether we re-*

Alg_Init	1000 points	4000 points	7000 points	10000 points	25000 points
ENT_FL	57.8 ± 0.0	78.6 ± 0.7	85.8 ± 0.8	88.2 ± 0.8	93.5 ± 0.2
ENT_Rand	53.7 ± 0.0	79.1 ± 0.3	85.9 ± 0.4	89.0 ± 0.3	93.6 ± 0.2
BADGE_FL	57.8 ± 0.0	77.9 ± 0.3	85.0 ± 0.7	88.5 ± 0.5	93.2 ± 0.4
BADGE_Rand	53.8 ± 0.0	78.1 ± 0.7	84.9 ± 0.4	88.6 ± 0.1	93.4 ± 0.4

Table 2: Test accuracy and standard deviation on CIFAR-10 across different seed set initializations. We find that the initial benefit of the carefully constructed seed sets vanishes after only a few rounds.

*set or update after a few rounds* (there is not a statistically significant difference in test accuracy with  $\alpha = 0.05$  between resetting or updating for any given strategy in the later rounds). A critical difference between our setup and that of [3] is that the latter does not use data augmentation. Furthermore, the latter employs Adam instead of SGD. In Appendix E, we report an experiment identical to the one conducted in this section that employs Adam but without data augmentation. In this setting, we find that updating the model is consistently worse than resetting the model in every round. This illustrates another interesting benefit of data augmentation: we do not significantly lose generalization performance if we choose to update the model, which can save compute resources for expensive training tasks. In Appendix E, we illustrate this point on other data sets.

**4.8 Scalability, Running Times, and Compute.** Next, we discuss the running times of AL. Repeating the baseline, we run AL with an AL batch size of 3k instances for 8 rounds on CIFAR-10 up to a total of 25k instances (half of CIFAR-10). The total time taken for the intermediate training (across rounds) is 8 hours; in contrast, the total time taken by BADGE selection (across rounds) is 1 hour whereas entropy sampling takes 3 minutes. The training time almost triples when we run for the entire data set. Specifically, the increase in training time over previous work can largely be attributed to data augmentation and SGD, which causes slower model convergence. This further implies increased energy consumption and CO<sub>2</sub> emissions. To help alleviate and reduce the training times and compute, we suggest a few approaches. First, we can increase the AL batch size, thereby reducing the number of training rounds (which we have shown to be a significant computational bottleneck). Secondly, we can update the model from the previous rounds. This has negligible impact on performance in later rounds (as shown in Sec. 4.7) but only speeds up the running time by 1.07 $\times$ . As a faster training strategy, we also examine the use of data subset selection via the recently proposed GRAD-MATCH [15] algorithm. Using only 30% of the labeled data for training, we see that there is negligible loss in accuracy while achieving a further 3 $\times$  speedup in training. Finally, since entropy sampling is 20 $\times$  faster than BADGE, we propose the use of entropy sampling unless the data set has redundancy and repetitions. More details on the timings are in Appendix F.

## 5 Conclusion

In this work, we study AL in the context of recent deep learning advances like data augmentation and so forth. We provide several insights for AL practitioners. Summarily: **1)** It is beneficial to use a training loop which applies data augmentation, SGD, and other AL generalization approaches to improve test accuracy and labeling efficiency. **2)** We study the labeling efficiency of AL across multiple image classification data sets. **3)** Sophisticated diversity-based approaches like BADGE make sense only if the data has a lot of redundancy; uncertainty sampling is otherwise good enough and more compute-efficient. **4)** Having more examples per class in the unlabeled set leads to higher labeling efficiencies from AL. **5)** Reasonable choices of the AL batch size and seed set do not affect AL performance. **6)** Updating the model from the previous round is comparable to retraining from scratch in the later AL rounds. **7)** Approaches like model updating and data subset selection can significantly reduce the turnaround time and carbon emissions from AL. The last point is critical because AL generalization using data augmentation and SGD can significantly decelerate training convergence to get desirable accuracy gains. We believe that model updating and data subset selection can significantly reduce these compute costs while maintaining comparable accuracy. As future work, we plan to expand our study of AL evaluation beyond image classification tasks via our [DISTIL](#) toolkit, analyzing the same facets in the context of object detection, NLP, and so forth.

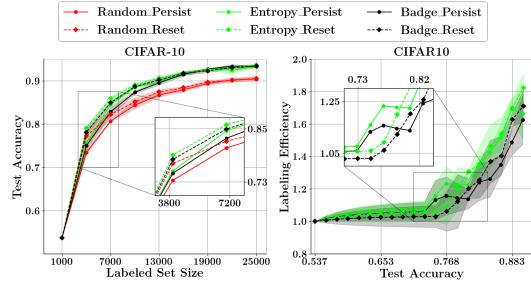


Figure 6: Reset v. Update. Resetting the model after each AL round achieves better initial performance, but both strategies converge later on.

## References

- [1] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.
- [2] Jordan Ash. Deep active learning with badge. <https://github.com/JordanAsh/badge>, 2019.
- [3] Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in Neural Information Processing Systems*, 33, 2020.
- [4] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2019.
- [5] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.
- [6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.
- [7] Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.
- [8] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [10] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.
- [11] Kuan-Hao Huang. Deep active learning. <https://github.com/ej0c16/deep-active-learning>, 2019.
- [12] Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pages 722–754. PMLR, 2021.
- [13] Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.
- [14] Vishal Kaushal, Rishabh Iyer, Suraj Kothawade, Rohan Mahadev, Khoshrav Doctor, and Ganesh Ramakrishnan. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1289–1299. IEEE, 2019.
- [15] Krishnateja Killamsetty, Durga Sivasubramanian, Baharan Mirzasoleiman, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: A gradient matching based data subset selection for efficient learning. In *ICML*, 2021.
- [16] Krishnateja Killamsetty, Durga Subramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: A generalization based data selection framework for efficient and robust learning. In *AAAI*, 2021.
- [17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
- [20] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

- [21] Prateek Munjal, Nasir Hayat, Munawar Hayat, Jamshid Sourati, and Shadab Khan. Towards robust and reproducible active learning using neural networks. *ArXiv*, *abs/2002.09564*, 2020.
- [22] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.
- [23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [24] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.
- [25] Jacob M Schreiber, Jeffrey A Bilmes, and William Stafford Noble. apricot: Submodular selection for data summarization in python. *J. Mach. Learn. Res.*, 21:161–1, 2020.
- [26] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.
- [27] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.
- [28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [29] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019.
- [30] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1954–1963, Lille, France, 07–09 Jul 2015. PMLR.
- [31] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4151–4161, 2017.
- [32] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [33] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [34] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.
- [35] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven HOI, and Weinan E. Towards theoretically understanding why sgd generalizes better than adam in deep learning. In *Proceedings of Neural Information Processing Systems, NeurIPS*, 2020.

# Supplementary Material for Effective Evaluation of Deep Active Learning on Image Classification Tasks

## Table of Contents

---

<b>A Reproducibility and Experimental Details</b>	<b>13</b>
A.1 Provided Experiments . . . . .	13
A.2 Code Repositories and Licenses . . . . .	14
A.3 Experimental Parameters . . . . .	14
A.4 MNIST Model Architecture . . . . .	16
<b>B Additional Baselines</b>	<b>16</b>
B.1 MNIST Baselines . . . . .	16
B.2 MNIST Low-Data Baselines . . . . .	17
B.3 Additional CIFAR-10 and SVHN Baselines . . . . .	18
<b>C Other Generalization Techniques</b>	<b>18</b>
<b>D Additional Redundancy Experiment</b>	<b>19</b>
<b>E Resetting with Adam and No Augmentation</b>	<b>19</b>
<b>F Scalability and Timing/Compute Analysis</b>	<b>20</b>
<b>G Significance Tests</b>	<b>21</b>
<b>H Societal Impacts and Limitations</b>	<b>22</b>

---

## A Reproducibility and Experimental Details

### A.1 Provided Experiments

In [DISTIL](#), we provide our experiments as Jupyter notebooks that are intended to be executed on Google Colab GPU VMs. In our executions, we utilized Colab Pro benefits in procuring these VMs. These VMs typically feature one 16 GB VRAM variant of the Nvidia P100 or V100 GPU, 12 GB to 25 GB of RAM, and two Intel(R) Xeon(R) CPUs. By using Google Colab VMs, we provide a widely accessible means of easily reproducing our results. Indeed, running each experiment amounts to running each cell within the Google Colab VM.

In our experiments, we also strive to use the same initial model whenever possible. In most of the provided Jupyter notebooks, we provide a configurable option to train a new initial model or to download and use a model from a Google Drive link. By default, we configure each experiment to train a new initial model. To then use that initial model in future runs, we recommend storing the model in a different location in Google Drive and downloading it from a shareable link as previously described.

Lastly, our notebooks provide a means of checkpointing since Colab VMs are not indefinitely kept alive. If an experiment is interrupted in this fashion, then simply running the preparation cells followed by the desired experiment cell should resume the experiment from the last checkpoint.

## A.2 Code Repositories and Licenses

Our experiments utilize our previously mentioned unified framework that draws upon contributions from existing sources and data sets. The following repositories and data sets are used, and their licenses are also provided:

- PyTorch [23]: Modified BSD
- Deep Active Learning [11]: None Listed
- BADGE [2]: None Listed
- VAAL [29]: BSD 2-Clause License
- DeepFool [20]: None Listed
- Apricot [25]: MIT License
- CIFAR-10 [18]: MIT License
- CIFAR-100 [18]: MIT License
- MNIST [19]: Creative Commons Attribution-Share Alike 3.0
- FMNIST [32]: MIT License
- SVHN [22]: CC0 1.0 Public Domain

## A.3 Experimental Parameters

Here, we list in detail the parameters used in each experiment. In our tables, **Max Epoch** refers to the maximum epoch allowed during training, **LR** refers to the learning rate used in training, **Opt. B. Size** refers to the batch size used by the optimizer, **Train Cut** refers to the training accuracy cutoff, **Opt.** refers to the optimizer, **AL B. Size** refers to the AL batch size, and **Seed Size** refers to the labeled seed set size.

**Baselines.** In Table 3, we provide the parameters for our baseline settings. Each row represents a data set and model architecture pair. For MNIST, our model architecture described in the next section is used. Note that the low-data MNIST setting describes the parameters for all strategies except for VAAL [29]. The training loop for VAAL is necessarily different as three models are trained; hence, we use the training loop provided in the VAAL repository. Also, we opt to use the default parameters specified in the VAAL repository mentioned previously. Additionally, we add functionality to support MNIST to the VAAL training loop. Lastly, CIFAR-100 uses a partitioned version of BADGE wherein the unlabeled set is divided into chunks. Each chunk is then handled by BADGE individually, and the chosen instances for each chunk are then merged into the final queried set of instances.

Baselines	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
CIFAR-10 ResNet-18	300	0.01	20	0.99	SGD	3000	1000
CIFAR-10 VGG-19	300	0.01	128	0.99	SGD	3000	1000
CIFAR-10 DLA	300	0.01	20	0.99	SGD	3000	1000
CIFAR-100 ResNet-18	300	0.01	20	0.99	SGD	5000	5000
MNIST	300	0.01	20	0.99	SGD	1000	300
MNIST Low	3000	0.01	20	0.99	SGD	10	50
FMNIST ResNet-18	150	0.01	64	0.99	SGD	1000	1000
SVHN ResNet-18	150	0.01	64	0.99	SGD	1000	1000
SVHN VGG-19	300	0.001	32	0.99	SGD	1000	1000

Table 3: Experimental parameters used in baseline comparisons. Note: Our baselines use VGG-19, not VGG-11. This is a typo in the main sections of the paper.

**Data Augmentation.** The experiments on data augmentation in Sec. 4.2 use the following parameters in Table 4. The VGG-11 runs specifically use Adam to match the setting in [4]. Note that an experiment with augmentation and an experiment without augmentation are run for each parameter configuration given in Table 4.

Exp.	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
CIFAR-10 ResNet-18	300	0.01	20	0.99	SGD	3000	1000
CIFAR-10 VGG-11	300	0.001	20	0.99	Adam	3000	1000

Table 4: Experimental parameters used in our data augmentation setting.

**Optimizer and Other Generalization Techniques.** The experiments on the choice of optimizer and the effect of other generalization techniques use the following parameters in Table 5. Each SS configuration uses a shake-shake-regularized ResNet-18. Each SWA configuration uses stochastic weight averaging. All other configurations use ResNet-18. All experiments are conducted on CIFAR-10.

Exp.	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
SGD	300	0.01	20	0.99	SGD	3000	1000
Adam	300	0.01	20	0.99	Adam	3000	1000
SWA	300	0.01	20	0.99	SGD/SWA	3000	1000
SS	300	0.01	20	0.99	SGD	3000	1000
SWA_SS	300	0.01	20	0.99	SGD/SWA	3000	1000

Table 5: Experimental parameters used in our optimizer and other generalization technique experiments.

**Redundancy.** The experiments on the effect of redundancy use the following parameters in Table 6. Note that both experiments (duplicating and augmenting) follow the strategy in **Sec. 4.4**, and the following parameter configurations are applicable for each amount of redundancy.

Exp.	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
Duplicate	300	0.01	20	0.99	SGD	500	500
Augment	300	0.01	20	0.99	SGD	500	500

Table 6: Experimental parameters used in our redundancy setting.

**Effect of Examples Per Class.** The experiments on the effect of examples per class in the unlabeled set use the following parameters in Table 7. As before, each experiment follows the strategy in **Sec. 4.5**, and the following parameter configurations are applicable for each number of examples per class in the unlabeled set.

Exp.	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
CIFAR-10	300	0.01	20	0.99	SGD	1000	1000
FMNIST	150	0.01	64	0.99	SGD	1000	1000

Table 7: Experimental parameters used in our examples per class experiment.

**AL Batch Sizes and Initializations.** The experiments on the effect of varying AL batch sizes and seed set initializations use the following parameters in Table 8. Each experiment uses CIFAR-10 and a ResNet-18 model.

Exp.	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
Bud_1000	300	0.01	20	0.99	SGD	1000	1000
Bud_3000	300	0.01	20	0.99	SGD	3000	1000
Bud_6000	300	0.01	20	0.99	SGD	6000	1000
Rand_Seed	300	0.01	20	0.99	SGD	3000	1000
FL_Seed	300	0.01	20	0.99	SGD	3000	1000

Table 8: Experimental parameters used in our AL batch size and seed set initialization settings. Note: FL refers to facility location.

**Model Reset vs. Update.** The experiments on the effect of model resetting versus model updating use the following parameters in Table 9.

Exp.	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
CIFAR-10 Reset	300	0.01	20	0.99	SGD	3000	1000
CIFAR-10 Update	300	0.01	20	0.99	SGD	3000	1000
CIFAR-10 N.A. Reset	300	0.01	20	0.99	Adam	3000	1000
CIFAR-10 N.A. Update	300	0.01	20	0.99	Adam	3000	1000
SVHN Reset	150	0.01	64	0.99	SGD	1000	1000
SVHN Update	150	0.01	64	0.99	SGD	1000	1000
SVHN N.A. Reset	150	0.01	64	0.99	Adam	1000	1000
SVHN N.A. Update	150	0.01	64	0.99	Adam	1000	1000

Table 9: Experimental parameters used in our reset versus update settings. N.A. represents an experiment where no augmentation is used.

**DSS.** The experiment using GRADMATCH [15] uses the following parameters in Table 10. Again, only 30% of the data is selected by GRADMATCH [15] every 35 epochs.

Exp.	Max Epoch	LR	Opt. B. Size	Train Cut	Opt.	AL B. Size	Seed Size
DSS CIFAR-10	300	0.01	20	0.99	SGD	3000	1000

Table 10: Experimental parameters used in our GRADMATCH [15] experiment.

#### A.4 MNIST Model Architecture

Here, we describe the model architecture discussed in Sec. 4. Our architecture performs the following operations on an input image:

1. 2-D convolution on image
2. ReLU on output of (1)
3. 2-D Convolution on output of (2)
4. ReLU on output of (3)
5. 2-D max pool on output of (4)
6. 2-D dropout on output of (5)
7. Fully connected layer on output of (6)
8. ReLU on output of (7)
9. 2-D dropout on output of (8)
10. Fully connected layer on output of (9)

Hence, our model architecture follows from LeNet [19], where two convolutional layers, max pooling, ReLUs, and final fully connected layers are used.

## B Additional Baselines

In this section, we provide additional baseline results on other combinations of data sets and model architectures. Namely, we provide our MNIST [19] baselines; the low-data MNIST baselines to compare BALD [10], VAAL [29], and adversarial DeepFool [7]; additional CIFAR-10 baselines using a VGG-19 [28] model and a DLA [33] model; and additional SVHN baselines using a VGG-19.

### B.1 MNIST Baselines

The results of our MNIST baseline are given in Figure 7. Of all our baselines, we achieve the highest labeling efficiencies in MNIST. As with the other baselines, the uncertainty-based strategies are competitive with BADGE [4].

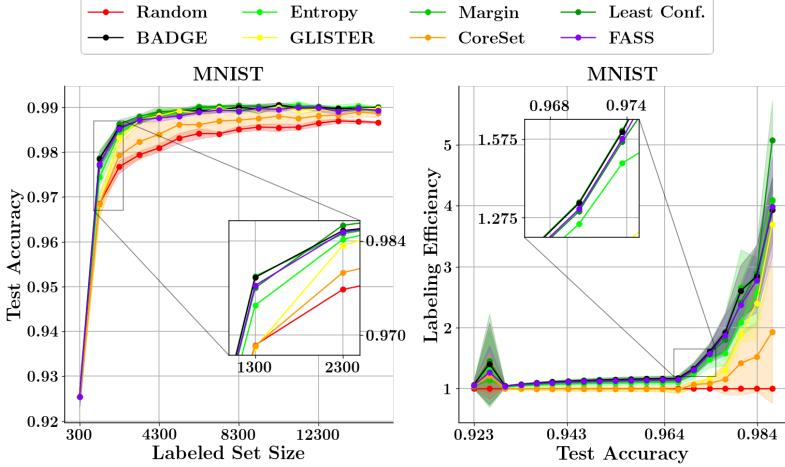


Figure 7: Baseline results with MNIST using our MNIST model architecture.

## B.2 MNIST Low-Data Baselines

To compare other recent AL strategies that did not scale well in our setting, we rerun our MNIST baseline using small budgets and low labeled set sizes. Namely, we analyze the performance of BALD [10], VAAL [29], and adversarial DeepFool [7]. In this baseline experiment, we use the code provided by each work whenever applicable in our setting. The results of our baseline experiment in this setting are given in Figure 8. We see that the only strategy (among the ones not considered in the main paper) to perform significantly well in our baseline setting is adversarial DeepFool [7].

VAAL [29] possibly does not do well in our setting because there is not enough time to train the VAE or the discriminator models before the task model reaches 0.99 training accuracy. Furthermore, BALD [10] does not perform well against the other strategies.

Lastly, we provide the average selection times for each AL strategy for this experiment in Table 11. We see that three of the four strategies not considered in the main sections have astronomically higher selection times compared to the other AL strategies. While VAAL has a low selection time, it requires  $4 \times$  time spent in training compared to the other strategies. This is mainly because in VAAL, we train two auxiliary models – the VAE and the discriminator – along with the main classification model. To be concrete, take the example of CIFAR-10. The cumulative training time of AL on CIFAR-10 with a AL batch size of 3000 and with 8 rounds of AL

MNIST Low	Average Selection Time
RANDOM	0.001
ENTROPY	15.482
GLISTER	62.685
BADGE	23.277
FASS	62.415
LEAST CONF	15.485
MARGIN	16.495
CORESET	26.539
BALD	528.519
VAAL	34.218
A. DEEPFOOL	1587.458

Table 11: Selection times for each AL strategy in each selection round (in seconds). The four strategies not compared in the main sections take much longer to select instances than the others. VAAL does not take that much time, but it requires approximately  $4 \times$  the time spent in training.

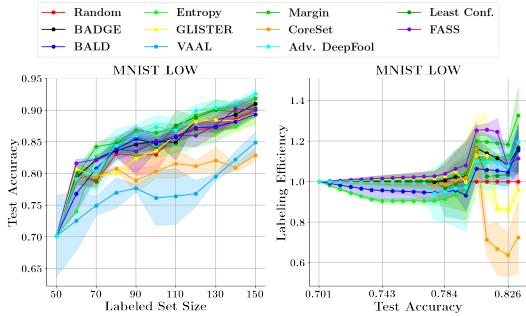


Figure 8: Baseline results with MNIST using our MNIST model architecture on very low set sizes. We see that adversarial DeepFool [7] performs the best out of the newly added strategies.

is 8 hours for most AL algorithms. In the case of VAAL, this cumulative time will be approximately 32 hours!

To conclude, we observe that VAAL, BALD, and adversarial DeepFool take very long for either selection or model training. On the other hand, there is not a significant gain in terms of performance (at least in the case of MNIST with small labeled set sizes), which suggests that entropy sampling and BADGE are probably better algorithms for AL in most cases.

### B.3 Additional CIFAR-10 and SVHN Baselines

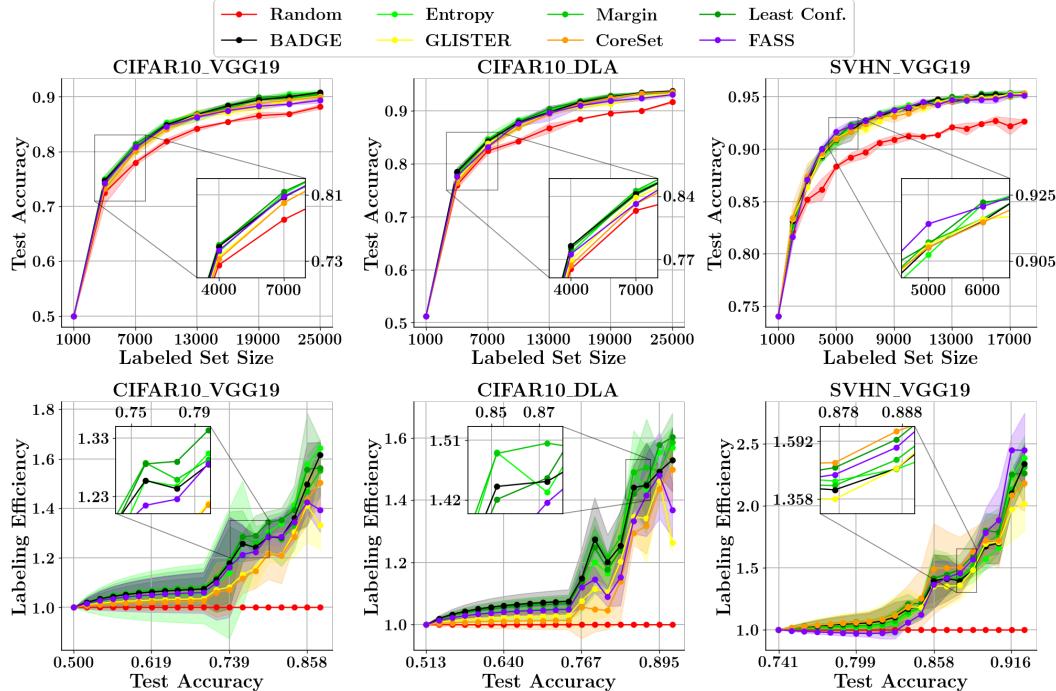


Figure 9: Additional baseline results for CIFAR-10 using VGG-19 [28] and DLA and for SVHN using only VGG-19. Largely, the takeaways remain the same.

To introduce other model architectures into our baseline comparisons for robustness, we repeat the same experimental procedure for a VGG-19 [28] model and a DLA [33] model for CIFAR-10 [18]. For SVHN [22], we only repeat with a VGG-19 model. The results of these experiments are given in Figure 9. Largely, the takeaways remain the same: For CIFAR-10, the labeling efficiency nears  $1.6\times$  (for both VGG and DLA), and the uncertainty-based AL techniques perform well against state-of-the-art techniques such as BADGE [4]. While the labeling efficiency is slightly reduced for SVHN, the labeling efficiency is still appreciably high (nearing  $2.5\times$ ), and the uncertainty-based AL techniques perform well against state-of-the-art techniques such as BADGE [4].

## C Other Generalization Techniques

In this section, we provide further insights on the effect of other generalization approaches. The first of these techniques is stochastic weight averaging (SWA) [13], which computes the learned weights of a model as the average of weights periodically obtained via SGD. The second of these techniques is shake-shake (SS) regularization [8], which can be used to replace the standard sum of branches in existing residual networks with a stochastic affine combination of the branches. Both techniques have empirically shown to improve generalization performance, and the effect of each on AL performance is studied here. In this experiment, we use the same setting as the CIFAR-10 baseline experiment; however, the SWA [13] experiments use a SWA optimizer after reaching 0.95 training accuracy, and the shake-shake regularization [8] experiments use a shake-shake-regularized residual network.

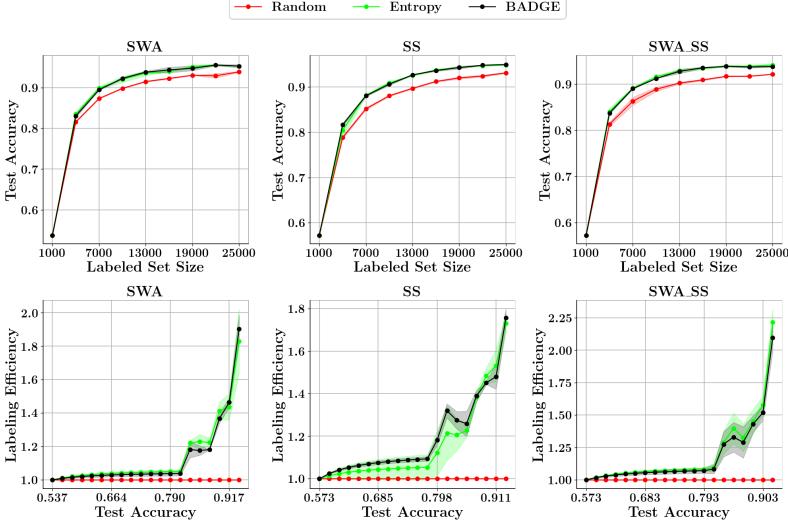


Figure 10: Effect of SWA and shake-shake (SS) regularization on our baseline settings. We see that both increase the generalization performance and do not negatively affect the labeling efficiencies.

The results of this experiment in Figure 10 indicate that both contribute to increasing the generalization performance of the model while maintaining high labeling efficiencies. Hence, we conclude that *the generalization techniques studied here positively contribute to the performance of AL algorithms & their labeling efficiency*, further enhancing the need for generalization approaches as stated in **P4**. The same observations (as seen in **Sec. 4.2**) carry forward: **a)** With the use of these generalization approaches, we see improvements in the test accuracy. **b)** We see an improvement in the labeling efficiency – in fact, with both SWA and SS, we see a labeling efficiency of up to  $2.25 \times$ , which is higher than the labeling efficiency we saw for the baseline experiments and also for SWA and SS applied individually. **c)** Entropy sampling and BADGE have similar performance – in fact, entropy sampling slightly outperforms BADGE both in performance and labeling efficiency with SWA and SS.

## D Additional Redundancy Experiment

In this section, we further explore the effect of redundancy on the performance of AL in our setting. The previous redundancy experiment explored the effect of duplicating instances from CIFAR-10 [18]; however, it is often not the case that instances are simply repeated in data sets. To study a more compelling redundancy scenario, we instead formulate redundant points by applying slight random translations and random cropping to each image. The rest of the parameters of the redundancy experiment are kept the same. Largely, we see that results remain the same: Diversity-based methods such as BADGE [4] are robust against redundant data while uncertainty-based methods such as entropy sampling [27] poorly handle redundant data. The results are shown in Figure 11.

## E Resetting with Adam and No Augmentation

In our initial experiments, we compared model updating and model retraining in the scope of SGD and data augmentation. However, to corroborate past work done in [3], we consider the impact of using Adam and no data augmentation on the performance degradation between model updating and model retraining. In this case, we repeat our updating *vs.* retraining experiment where we instead use Adam and no data augmentation. From our results in Figure 12, we see that the performance degradation of updating the model after each AL selection round *vs.* resetting the model after each AL selection round is markedly larger, especially in CIFAR-10. Hence, we see that updating the previous model hurts generalization performance when using Adam and no augmentation, which confirms the warm-starting generalization deficiency stated in [3]. We see that in our setting, however, that the generalization gap is significantly reduced in both data sets.

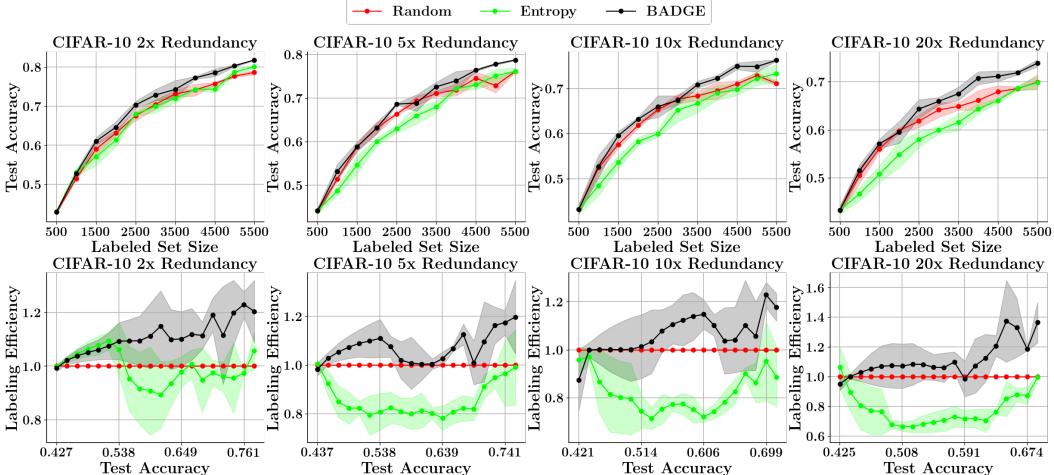


Figure 11: Performance of AL strategies on a redundant CIFAR-10 where the redundancy comes from subtle image augmentations. The takeaways are the same as in the first redundancy experiment.

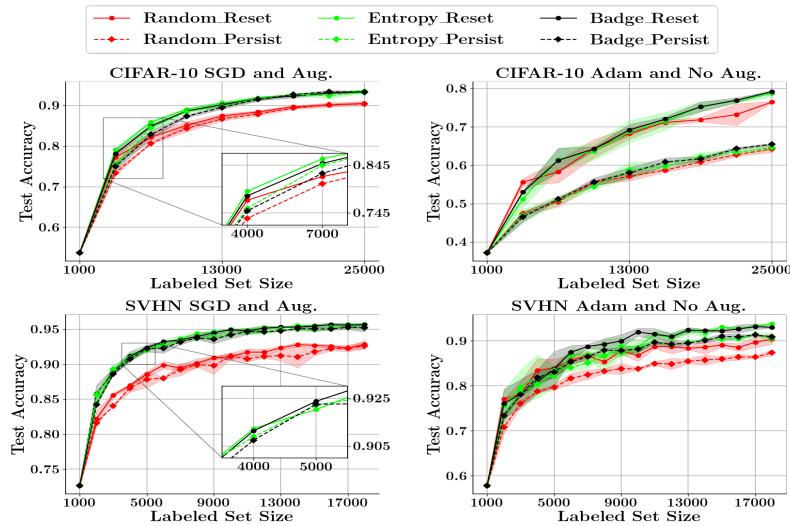


Figure 12: Test accuracies of SVHN and CIFAR-10 model resetting vs. model updating. Here, we also provide an identical setting where Adam and no augmentation are used. We see that the results from [3] are supported by the clear performance gap in the right plots. **Note: "Persist" is synonymous with updating as the model persists across AL rounds.**

## F Scalability and Timing/Compute Analysis

We briefly discussed the summary of the compute times used in our baseline experiments for BADGE and entropy sampling. Here, we provide in Table 12 the selection times and training times of that experiment. We see that the time spent in the AL loop is largely dominated by the intermediate model training and that entropy sampling takes significantly less time than BADGE to select unlabeled instances. Specifically, the cumulative training time is around 8 hours. The cumulative selection time for BADGE is close to one hour; in contrast, it takes only 3 minutes for entropy sampling. We also see that there is not a significant reduction if we do model updating instead of model retraining (Table 12). In particular, the cumulative training time only reduces to 7.6 hours.

To examine other methods by which we can accelerate these training times, we conduct an experiment where most of the intermediate training rounds use adaptive DSS techniques such as GRAD-MATCH. To elicit the true intermediate model performance across the AL loop, we periodically use full training in a few of the AL rounds. We present in Table 13 the test accuracies obtained from our GRADMATCH experiment during these full training rounds. While the test accuracies in the intermediate rounds were lower than what were reported in the CIFAR-10 baseline setting (not

shown), we see that the accuracy is maintained in full training rounds. The reason for this is that there is a drop in accuracy of about 1% with the use of an approach like GRADMATCH, but using the slightly sub-optimal model for selecting the data points does not yield a significant drop in accuracies. Hence, there is almost no difference in performance in the rounds where full training is done. Thus, we need to interleave full training to periodically gauge the performance of AL. Furthermore, we note that the intermediate training rounds complete roughly  $3\times$  faster with GRADMATCH using 30% subset sizes; hence, it makes sense to use GRADMATCH in some of the intermediate training rounds to accelerate AL. Applying these speedups to the entropy sampling times obtained in Table 12 for the applicable columns, we see that the AL loop using GRADMATCH reduces to 5 hours, resulting in a  $1.6\times$  speedup. We note that this speedup factor is reduced due to the full training rounds; hence, AL loops that perform full training more infrequently can appreciate larger speedups.

Timing	R1	R2	R3	R4	R5	R6	R7	R8
Entropy (Reset)	609.2	1397.5	2156.6	3087.2	3975.2	4940.7	5767.0	6711.6
Entropy (Update)	498.8	1212.7	2118.0	3039.7	3823.7	4689.5	5531.0	6520.7
BADGE (Reset)	613.1	1339.1	2120	3009.7	3942.3	4880.8	5753.3	6450.5
BADGE (Update)	417.7	1083.1	1874.6	2862.6	3691.3	4540.6	5353.2	6224.2
Entropy (Sel.)	32.3	30.9	28.4	26.2	24.2	22.4	20.6	18.4
BADGE (Sel.)	620.5	572.6	535.9	495.3	456.9	419.3	386.6	343.1

Table 12: Time spent in training and selection (seconds) for BADGE and entropy sampling in our CIFAR-10 baseline. The first two rows show the training times spent by entropy sampling and BADGE when using model updating and model resetting. The third row shows the time spent by each AL strategy during their selection phases.

DSS	1000 points	10000 points	19000 points	25000 points
RANDOM (GM)	$53.7 \pm 0.0$	$85.2 \pm 0.4$	$89.7 \pm 0.3$	$90.5 \pm 0.5$
RANDOM (F)	$53.7 \pm 0.0$	$85.1 \pm 0.9$	$89.7 \pm 0.4$	$90.5 \pm 0.6$
ENTROPY (GM)	$53.7 \pm 0.0$	$88.5 \pm 0.1$	$92.6 \pm 0.4$	$93.5 \pm 0.5$
ENTROPY (F)	$53.7 \pm 0.0$	$89.0 \pm 0.3$	$92.6 \pm 0.1$	$93.6 \pm 0.2$
BADGE (GM)	$53.7 \pm 0.0$	$87.9 \pm 0.5$	$92.2 \pm 0.6$	$93.3 \pm 0.6$
BADGE (F)	$53.7 \pm 0.0$	$88.7 \pm 0.1$	$92.3 \pm 0.4$	$93.4 \pm 0.4$

Table 13: GRADMATCH at full training rounds vs. baseline setting at those rounds for CIFAR-10. GRADMATCH maintains comparable accuracy while being significantly faster in the other rounds not shown.

## G Significance Tests

The significance tests done on the results of our experiments are modeled after the penalty matrix computation done in [4]. Namely, the significance results reported in the main sections of this paper are derived off the cells of each matrix. For clarity, the process of computing these matrices is described, and explanations are given for the claims made in the paper.

The pairwise penalty matrices (PPMs) computed in this section follow the strategy used in [4]. In their strategy, a penalty matrix is constructed for each data set and model pair. Each cell  $(i, j)$  of the matrix reflects the fraction of training rounds that AL with selection algorithm  $i$  has higher test accuracy than AL with selection algorithm  $j$  with statistical significance. As such, the average difference between the test accuracies of  $i$  and  $j$  and the standard error of that difference are computed for each training round. A two-tailed  $t$ -test is then performed for each training round: If  $t > t_\alpha$ , then  $\frac{1}{N_{train}}$  is added to cell  $(i, j)$ . If  $t < -t_\alpha$ , then  $\frac{1}{N_{train}}$  is added to cell  $(j, i)$ . Hence, the full penalty matrix gives a holistic understanding of how each selection algorithm compares against the others: A row with mostly high values signals that the associated selection algorithm performs better than the others; however, a column with mostly high values signals that the associated selection algorithm performs worse than the others.

In our case, we use  $\alpha = 0.05$  for our  $t$ -tests. The relevant matrices are computed for each of the following claims, and explanations for each claim are also given:

- **Claim 1:** Entropy sampling performs better than BADGE in just one of the selection rounds with  $\alpha = 0.05$  (**Sec 4.2**). In Figure 13a, cell (ENTROPY,BADGE) reads 0.11, and there were only 9 training rounds, so entropy sampling does better than BADGE in just one round. In all other rounds, there is no statistically significant difference between the two.

- **Claim 2:** Entropy sampling with SGD scores higher than with Adam in all rounds except the initial round with  $\alpha = 0.05$  (**Sec 4.3**). In Figure 13b, cell (ENTROPY\_SGD, ENTROPY\_ADAM) reads 0.89, and there were only 9 training rounds, so entropy sampling with SGD does better than entropy sampling with Adam in all but one round. Since the initial accuracy is the same across each configuration, it must be the initial round.
- **Claim 3:** No two configurations of the AL batch size had accuracies significantly different with  $\alpha = 0.05$  for more than 20% of the selection rounds (**Sec 4.6**). In Figure 14a, all cells are a maximum of 0.2, which confirms that the 5 comparable rounds (1000, 7000, 13000, 19000, 25000) only had significant differences in 20% of the rounds.
- **Claim 4:** Entropy sampling with a random initialization versus entropy sampling with a facility location initialization is the only comparison to show a significant difference in test accuracy in a selection round with  $\alpha = 0.05$  (**Sec 4.6**). As shown in Figure 14b, the only cell that is not zero is (ENTROPY\_RAND, ENTROPY\_FL); hence, the claim follows.
- **Claim 5:** There is not a statistically significance difference in test accuracy with  $\alpha = 0.05$  between resetting or updating for any given strategy in the later rounds (**Sec 4.7**). In Figure 15a, the first matrix shown is computed for the first four training rounds while the second matrix shown is computed for the last five training rounds. As shown, each (\*\_UPDATE, \*) cell in the latter matrix is 0, and so is each (\*, \*\_UPDATE) cell. Hence, the claim follows.

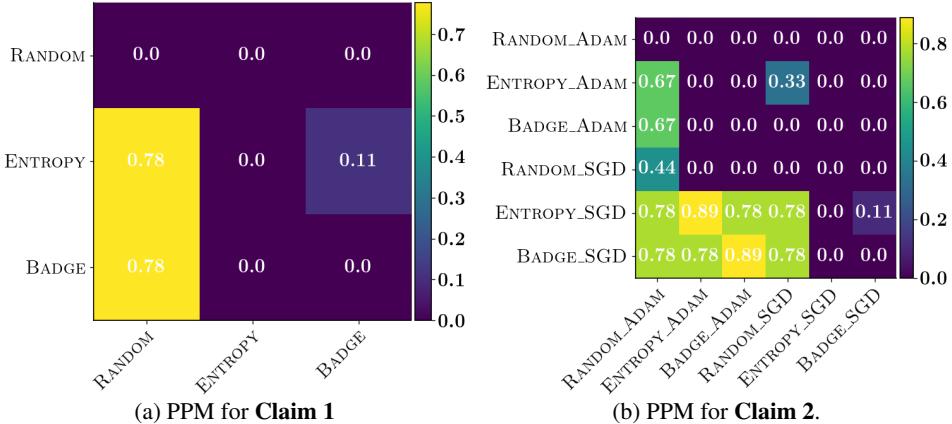
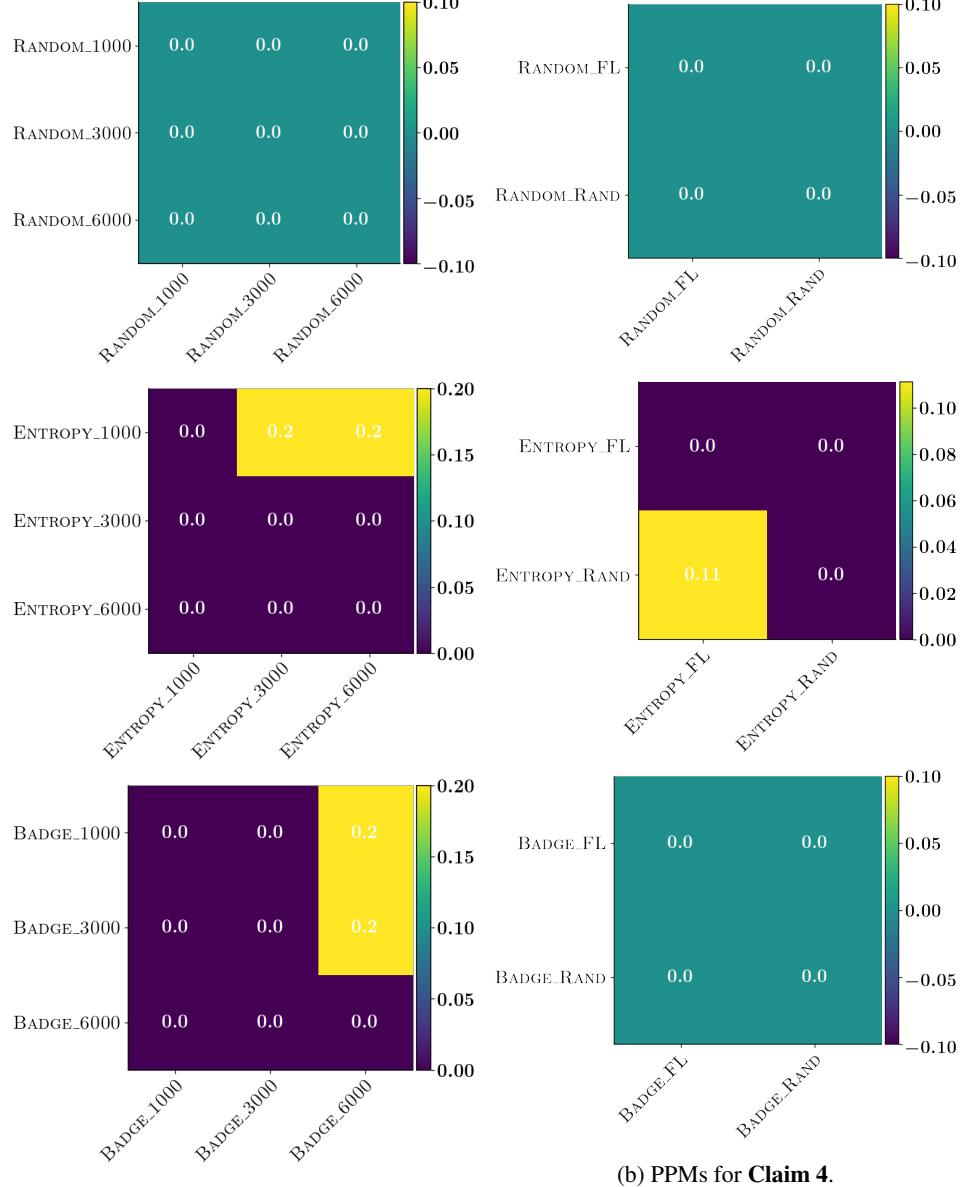


Figure 13: PPMs (pair-wise penalty matrices) for first two claims. The left shows that only one round of entropy sampling performs better than BADGE. The right shows that entropy sampling with SGD performs better than entropy sampling with Adam in all but one round.

## H Societal Impacts and Limitations

It is important to consider the limitations of this work and the potential societal impacts that this work has. Perhaps the largest limitation of this work is its reliance on data augmentation. Indeed, not all tasks in AL afford the use of data augmentation. In our case, we studied the effect of augmenting image data for classification tasks; however, many other AL domains such as text-based tasks do not have nearly the same capability of data augmentation. Hence, many of the observations in this paper only apply to domains that readily accept data augmentation, which typically applies to image data. Another limitation is that the results of this paper are prefaced on academic data sets and modifications thereof; applicability to other image classification data sets may not always follow the trends mentioned in this paper. However, we highly suspect that the trends mentioned in this paper apply to most image classification tasks where data augmentation can be done.

Although it is not a main talking point in this work, effectively evaluating AL algorithms does have societal impacts. Namely, not effectively evaluating AL algorithms can present misguided results about which AL algorithm performs well, which can have adverse effects on everything benefiting from AL tasks. As an example, since many AL tasks now operate on images of people, it is imperative for AL algorithms to achieve state-of-the-art generalization performance to prevent accidental discrimination. If an AL algorithm is not evaluated effectively using the generalization



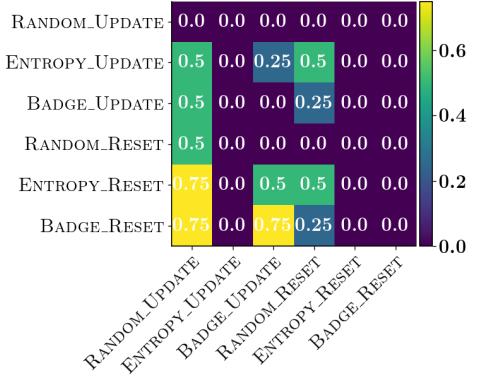
(a) PPMs for **Claim 3**.

(b) PPMs for **Claim 4**.

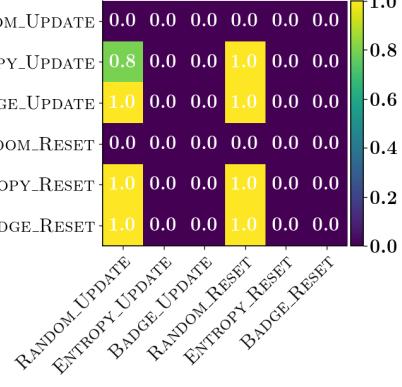
Figure 14: PPMs for the third and fourth claims for random sampling, BADGE, and ENTROPY. While it makes sense that there is no difference for random sampling, we see very little difference even for ENTROPY and BADGE across different batch sizes (left) and initializations (right). Note: Dmin refers to dispersion-min while FL refers to facility location.

techniques discussed in this paper, then its actual performance in industry might suffer, resulting in worse generalization performance compared to what could have been achieved using other AL methods. By effectively evaluating AL algorithms in our setting, we can ensure that the results on each AL algorithm are faithful to all the practices done in industry to achieve good generalization performance.

Finally, as discussed in the conclusion of the main paper, the suggestions for better generalization approaches (*e.g.*, the use of data augmentation and SGD in the training loop or even more complicated SWA/SS approaches) come with a significantly higher cost of compute. To achieve better generalization, we essentially need to train the models for more epochs, which results in higher experimental turn-around times, higher energy consumption, and higher CO<sub>2</sub> emissions. We hope that many of the suggestions here like the use of data subset selection and specifically GRADMATCH [15] can help in reducing the turn-around times, energy consumption, and CO<sub>2</sub> emissions.



(a) PPM for **Claim 5**.



(b) PPM for **Claim 5 (2)**.

Figure 15: The first PPM is calculated for the first four AL rounds; the second PPM is calculated for the remaining AL rounds. We see from the second matrix that there is no significant difference in test accuracies between updating and resetting for a given AL strategy.