A SIMPLE YET POWERFUL DEEP ACTIVE LEARNING WITH SNAPSHOTS ENSEMBLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Given an unlabeled pool of data and the experts who can label them, active learning aims to build an agent that can effectively acquire data to be queried to the experts, maximizing the gain in performance when trained with them. While there are several principles for active learning, a prevailing approach is to estimate uncertainties of predictions for unlabeled samples and use them to define acquisition functions. Active learning with the uncertainty principle works well for deep learning, especially for large-scale image classification tasks with deep neural networks. Still, it is often overlooked how the uncertainty of predictions is estimated, despite the common findings on the difficulty of accurately estimating uncertainties of deep neural networks. In this paper, we highlight the effectiveness of snapshot ensembles for deep active learning. Compared to the previous approaches based on Monte-Carlo dropout or deep ensembles, we show that a simple acquisition strategy based on uncertainties estimated from parameter snapshots gathered from a single optimization path significantly improves the quality of the acquired samples. Based on this observation, we further propose an efficient active learning algorithm that maintains a single learning trajectory throughout the entire active learning episodes, unlike the existing algorithms training models from scratch for every active learning episode. Through the extensive empirical comparison, we demonstrate the effectiveness of snapshot ensembles for deep active learning.

1 INTRODUCTION

The progress of deep learning is largely driven by data, and we often work with well-curated and labeled benchmark data for model developments. However, in practice, such nicely labeled data are rarely available. Many of the data accessible to practitioners are unlabeled, and more importantly, labeling such data incurs costs due to human resources involved in the labeling process. Active Learning (AL) may reduce the gap between the ideal and real-world scenarios by selecting the informative samples from the unlabeled pool of data, so after being labeled and trained with them, a model can maximally improve the performance.

The main ingredient of an active learning algorithm is the acquisition function which ranks the samples in an unlabeled pool with respect to their utility for improvement. While there are several possible design principles (Ren et al., 2021), in this paper, we mainly focus on the acquisition functions based on the uncertainty of the predictions. Intuitively speaking, given a model trained with the data acquired so far, an unlabeled example exhibiting high predictive uncertainty with respect to the model would be a "confusing" sample which would substantially improve the model if being trained with the label acquired from experts. A popular approach in this line is Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011), where a committee of multiple models predicts an unlabeled sample, and the degree of disagreement is measured as a ranking factor. Here, the multiple models are usually constructed in a Bayesian fashion, and their disagreement reflects the epistemic (model) uncertainty about the prediction. BALD is demonstrated to scale well for modern deep neural networks for high-dimensional and large-scale data (Gal et al., 2017).

Similar to BALD, many AL algorithms based on uncertainty employ a committee of models to estimate the epistemic uncertainty of the predictions. The problem is, for deep neural networks trained with high-dimensional data, it is often frustratingly hard to accurately estimate the epistemic uncertainty. Gal et al. (2017) propose to use Monte-Carlo DropOut (MCDO) (Gal and Ghahramani, 2017),

an instance of variational approximation to the posteriors and predictive, and similarly, Rakesh and Jain (2021) suggest using more generic spike-and-slab variational posteriors (Louizos et al., 2017). However, variational approximations tend to underestimate the posterior variances (Blei et al., 2017), so the uncertainty-based acquisition functions computed from them may be suboptimal. Alternatively, one can employ Deep Ensemble (DE) (Lakshminarayanan et al., 2017), where a single model is trained multiple times with the same data but with different random seeds for initialization and mini-batching. While being simple to implement, DE works surprisingly well, surpassing most of the Bayesian Neural Networks (BNN) alternatives both in terms of accuracy and predictive uncertainty (Fort et al., 2021; Ovadia et al., 2019). To this end, Beluch et al. (2018) highlighted the effectiveness of DE as a way to estimate uncertainty for acquisition functions and demonstrated excellent performance.

A downside of DE is that it is expensive as it requires multiple models to be trained and maintained for inference. As an alternative, a Snapshot Ensemble (SE) (Huang et al., 2017; Garipov et al., 2018) proposes to collect multiple model snapshots (checkpoints) within a single learning trajectory, rather than collecting them at the end of the multiple learning trajectories as in DE. Compared to DE, with SE, one can quickly construct a decent set of models without having to go over multiple training runs while not losing too much accuracy.

Inspired by the advantage of the SE, we study the use of the snapshot ensemble to AL. Specifically, we estimate the epistemic uncertainties from SE and use them to evaluate the acquisition functions. Through an extensive empirical comparison, we show that the AL based on SE significantly outperforms the existing approaches, even comparable to or better than the one with DE; this is somewhat surprising since it is often reported that SE is less accurate than DE (Ashukha et al., 2020). Moreover, based on this observation, we propose a novel active learning algorithm that can greatly reduce the number of training steps until the final acquisition. Typically, an AL algorithm alternates between acquiring labels based on a model and re-training the model with the newly acquired labels. Here, for every re-training step, the old models are discarded, and a new model is trained from scratch. Instead, we suggest maintaining a model on a single learning trajectory throughout the entire active learning procedure and gathering snapshots from the trajectory to compute acquisition functions. We show that this can significantly reduce the number of training steps without sacrificing too much accuracy. We summarize our contributions as follows:

- We propose to use SE for the uncertainty-based acquisition functions for AL and demonstrate its effectiveness through various empirical evaluations.
- We propose a novel AL algorithm where a single learning trajectory is maintained and used to compute acquisition functions throughout the entire AL procedure. We demonstrate that our algorithm could achieve decent accuracy with much fewer training steps.

2 Background

2.1 Settings and basic active learning algorithm

In this paper, we mainly discuss K-way classification problem, where the goal is to learn a classifier $f(\cdot;\boldsymbol{\theta})$, parameterized by $\boldsymbol{\theta}$, taking an input $\boldsymbol{x} \in \mathbb{R}^d$ to produce a K-dimensional probability vector, that is, $f(\boldsymbol{x};\boldsymbol{\theta}) \in [0,1]^K$ such that $\sum_{k=1}^K f_k(\boldsymbol{x};\boldsymbol{\theta}) = 1$. To learn $\boldsymbol{\theta}$, we need a labeled dataset consisting of pairs of an input $\boldsymbol{x}$ and corresponding label $y \in \{1,\dots,K\}$, but in AL, we are given only an unlabeled dataset $\mathcal{U} = \{\boldsymbol{x}_i\}_{i=1}^n$ without labels.

An AL algorithm is defined with a classifier model $f(\cdot;\boldsymbol{\theta})$ and an *acquisition function* $a : \mathbb{R}^d \to \mathbb{R}$ measuring how useful an unlabeled example $\boldsymbol{x}$ is to the classifier $f(\cdot;\boldsymbol{\theta})$. Given f and a, an AL algorithm alternates between acquiring the labels for chosen unlabeled samples and training the classifier with the labeled samples. A single iteration of acquiring samples and training the classifier is called an *episode*. In the first episode, m samples are randomly chosen from $\mathcal{U}$, and the labels are acquired for them to constitute an initial training set $\mathcal{D}_{\text{train}}$. The classifier is then trained with $\mathcal{D}_{\text{train}}$ to obtain $\boldsymbol{\theta}_1$, or in case of the uncertainty based AL, a set of parameters $\{\boldsymbol{\theta}_1^{(s)}\}_{s=1}^S$. The labeled samples are removed from $\mathcal{U}$. For all subsequent episodes $t \geq 2$, with the parameters $\{\boldsymbol{\theta}_{t-1}^{(s)}\}_{s=1}^S$ from the previous episode, the samples remaining in $\mathcal{U}$ are ranked with the values of the acquisition function, and the top m of them are selected to be labeled. The newly labeled m samples are then

appended to the labeled training set $\mathcal{D}_{\text{train}}$, and the classifier is trained *from scratch* with the extended $\mathcal{D}_{\text{train}}$ to obtain $\{\boldsymbol{\theta}_t^{(s)}\}_{s=1}^S$. The algorithm terminates when it reaches the predetermined number of episodes T, and the goal of AL is to maximize the accuracy of the classifier after the final episode.

2.2 Acquisition functions

Here, we review some popular choices for the acquisition functions, especially the ones based on predictive uncertainties. Uncertainty-based acquisition functions estimate the epistemic uncertainty of the prediction, and the epistemic uncertainty is represented via a set of parameters $\{\boldsymbol{\theta}^{(s)}\}_{s=1}^S$. This set of parameters defines a committee of models $\{f(\cdot;\boldsymbol{\theta}^{(s)})\}_{s=1}^S$.

Maximum Entropy (ME). ME measures the predictive entropy of a given example $\boldsymbol{x}$ which can be approximated as,

$$\begin{aligned} H[\boldsymbol{y}|\boldsymbol{x},\mathcal{D}_{\text{train}}] &= -\sum_{k=1}^K p(y=k|\boldsymbol{x},\mathcal{D}_{\text{train}}) \log p(y=k|\boldsymbol{x},\mathcal{D}_{\text{train}}) \\ &\approx -\sum_{k=1}^K \left(\frac{1}{S}\sum_{s=1}^S f_k\big(\boldsymbol{x};\boldsymbol{\theta}^{(s)}\big)\right) \log \left(\frac{1}{S}\sum_{s=1}^S f_k\big(\boldsymbol{x};\boldsymbol{\theta}^{(s)}\big)\right). \end{aligned} \tag{1}$$

Larger entropy means that the model is more uncertain about the prediction.

Variation Ratio (VR). VR (Freeman, 1965) measures how certain the model is about its prediction for $\boldsymbol{x}$, or how many of the committee agree with the prediction.

$$1-\max_k p(y=k|\boldsymbol{x},\mathcal{D}_{\text{train}}) \approx 1-\max_k \left(\frac{1}{S}\sum_{s=1}^S f_k\big(\boldsymbol{x};\boldsymbol{\theta}^{(s)}\big)\right). \tag{2}$$

Bayesian Active Learning by Disagreement (BALD). BALD (Houlsby et al., 2011) measures the mutual information between the label and the parameter given an input $\boldsymbol{x}$ and the training data $\mathcal{D}_{\text{train}}$. It can also be interpreted as measuring disagreement between the predictions of the committee members. BALD is maximized when each of the committee members is certain about their own predictions (small entropy $H[y|\boldsymbol{x},\boldsymbol{\theta}^{(s)}]$), but the predictions disagree with each other, so the averaged prediction becomes uncertain (high entropy $H[y|\boldsymbol{x},\mathcal{D}_{\text{train}}]$).

$$\begin{aligned} I[y,\boldsymbol{\theta}|\boldsymbol{x},\mathcal{D}_{\text{train}}] &= H[y|\boldsymbol{x},\mathcal{D}_{\text{train}}] - \mathbb{E}_{\boldsymbol{\theta}|\mathcal{D}_{\text{train}}}\big[H[y|\boldsymbol{x},\boldsymbol{\theta}]\big] \\ &\approx -\sum_{k=1}^K \left(\frac{1}{S}\sum_{s=1}^S f_k\big(\boldsymbol{x};\boldsymbol{\theta}^{(s)}\big)\right) \log \left(\frac{1}{S}\sum_{s=1}^S f_k\big(\boldsymbol{x};\boldsymbol{\theta}^{(s)}\big)\right) \\ &\quad + \frac{1}{S}\sum_{s=1}^S\sum_{k=1}^K f_k\big(\boldsymbol{x};\boldsymbol{\theta}^{(s)}\big) \log f_k\big(\boldsymbol{x};\boldsymbol{\theta}^{(s)}\big). \end{aligned} \tag{3}$$

2.3 Estimating epistemic uncertainty.

In Bayesian AL, the model parameter $\boldsymbol{\theta}$ is treated as a random variable with prior $p(\boldsymbol{\theta})$ and the posterior $p(\boldsymbol{\theta}|\mathcal{D}_{\text{train}})$ is approximated via a set of samples $\{\boldsymbol{\theta}^{(s)}\}_{s=1}^S$. There are several ways to approximate it.

Variational approximations. For variational approximation, an easy-to-handle variational distribution $q(\boldsymbol{\theta})$ is introduced, learned to minimize $D_{\text{KL}}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D}_{\text{train}})]$, and used as a proxy for $p(\boldsymbol{\theta}|\mathcal{D}_{\text{train}})$. That is, once obtained approximate distribution $q(\boldsymbol{\theta})$, we draw $\boldsymbol{\theta}^{(1)},\dots,\boldsymbol{\theta}^{(S)} \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{\theta})$. A popular choice for $q(\boldsymbol{\theta})$ is mean-field Gaussian distribution (Blundell et al., 2015). In AL literature, MCDO is widely used after (Gal et al., 2017), where the dropout (Srivastava et al., 2014) is applied to the model f and the randomness due to the dropout is interpreted as an approximate posterior $q(\boldsymbol{\theta})$. While relatively simple to implement, the variational approximations are known to underestimate the posterior variances.

Deep ensembles. DE (Lakshminarayanan et al., 2017) trains $f(\cdot;\boldsymbol{\theta})$ multiple times with the same $\mathcal{D}_{\text{train}}$ but with different random initializations to obtain $\{\boldsymbol{\theta}^{(s)}\}_{s=1}^{S}$. DE is simple to implement, and yet its performance is remarkable, achieving state-of-the-art across various applications. The power of DE is mainly from its property to pick parameters from multiple modes (Fort et al., 2021), so the committee constructed from them yields a diverse set of predictions. Even if it is not explicitly assuming the prior for $p(\boldsymbol{\theta})$, DE can roughly be interpreted as an approximate Bayesian inference method (Wilson and Izmailov, 2021; D'Angelo and Fortuin, 2021), so the parameters $\{\boldsymbol{\theta}^{(s)}\}_{s=1}^{S}$ can be interpreted as posterior samples approximating the epistemic uncertainty of the model.

Snapshot ensembles. DE is expensive, both for training and inference since it has to keep multiple models with different parameters. SE (Huang et al., 2017; Garipov et al., 2018) is an idea to reduce the training cost of DE, where the multiple parameters $\{\boldsymbol{\theta}^{(s)}\}_{s=1}^{S}$ are gathered within a *single* training run rather than collected from multiple training runs. To obtain diverse parameters, the learning rate of the training run is carefully chosen to encourage the optimization path to explore a wide area of the loss surface, and the parameter "snapshots" are periodically captured during the training run. The SE usually underperforms DE with the same number of parameters gathered, but it can collect the parameters much faster since it requires only a single training run.

3 METHODS

3.1 ACTIVE LEARNING WITH SNAPSHOT ENSEMBLES

We first present an AL algorithm based on SE. The idea is simple; during the classifier training stage of each episode, after a predefined number of steps, we periodically store the parameter snapshots with a learning rate higher than the training stage and use them to compute the acquisition functions at the end of the episode. Unlike the AL based on DE, SE does not incur any additional cost for training and acquisition. In the final episode, we have several options to choose from when we train the classifier using the acquired data $\mathcal{D}_{\text{train}}$ for the last time. As in vanilla AL, we may follow a single learning trajectory and pick the parameter at the last step as a point estimate. Alternatively, we can apply DE to get an ensembled model. Algorithm 1 summarizes the AL algorithm with acquisition functions based on SE. Note that we are obtaining the final classifier with the vanilla Stochastic Gradient Descent (SGD) in Algorithm 1, but we may instead apply DE for that. In Section 5, we show that this simple modification with SE, albeit no increase in the training and inference costs, significantly improves the performance, even outperforming the AL with DE.

3.2 ACTIVE LEARNING WITH SNAPSHOT ENSEMBLES AND FINE-TUNING

As described in Algorithm 1, the classifiers being trained for the intermediate episodes are used to compute the acquisition function and thrown away. However, considering that the acquisition function computed from a single learning trajectory work well, we may just keep a single learning trajectory *throughout the entire episodes* to collect snapshots on it. That is, whenever an episode terminates, instead of throwing away the classifier parameter, *use it as an initialization* for the next episode. We call this strategy Fine-tuning (FT) since the classifier training of each episode can be considered as a fine-tuning from the result of the previous episode. There are two things to note here. First, since we are continuously fine-tuning a single model, we need a much less number of training steps for each episode compared to the vanilla AL. Second, when evaluated at the intermediate episodes, the classifier may be less accurate than the classifiers being trained from scratch as in vanilla AL. This is somewhat expected, but after all, *what really matters is the accuracy of the classifier after the final episode*. We argue that the intermediate episodes are to acquire acquire samples quickly for the final training, so the accuracy of the classifier during that process is not important.

Also, our AL procedure with fine-tuning is reminiscent of continual learning, in the sense that a single model is continually being trained for multiple episodes having different data. To this end, we employ two commonly used tricks in continual learning literature.

Replay buffer. In principle, for the purpose of fine-tuning, we may use only the newly acquired data from the previous episode for fine-tuning. However, this would cause catastrophic forgetting

Algorithm 1: AL with SE

Input : Unlabeled dataset $\mathcal{U}$, number of episodes T, number of (acquisitions m, snapshots S, SGD steps N) per episode, acquisition function a, snapshot threshold steps N_{thres}, objective function $\mathcal{J}$, learning rate schedule η.
Ouput: A classifier $f(\cdot; \boldsymbol{\theta}_*)$.
Randomly draw m samples from $\mathcal{U}$, remove them from $\mathcal{U}$, and set them as $\mathcal{D}_{\text{train}}$.
for $t = 1, \ldots, T$ **do**
  Randomly initialize $\boldsymbol{\theta}_0$.
  Set $\boldsymbol{\Theta} \leftarrow \varnothing$.
  for $j = 1, \ldots, N$ **do**
    Draw a mini-batch $\mathcal{B}$ from $\mathcal{D}_{\text{train}}$.
    $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_{j-1} - \eta(j) \nabla_{\boldsymbol{\theta}} \mathcal{J}(\mathcal{B}, \boldsymbol{\theta}_{j-1})$.
    if $j \geq N_{\text{thres}} \wedge \mathrm{mod}\,(j - N_{\text{thres}}, \lfloor \frac{N - N_{\text{thres}}}{S} \rfloor) = 0$ **then**
      $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} \cup \{\boldsymbol{\theta}_j\}$.
    end
  end
  if $t < T$ **then**
    Compute $a(\boldsymbol{x}, \boldsymbol{\Theta})$ for all $\boldsymbol{x} \in \mathcal{U}$.
    Pick top m samples, remove them from $\mathcal{U}$, and append them to $\mathcal{D}_{\text{train}}$.
  else
    Set $\boldsymbol{\theta}_* \leftarrow \boldsymbol{\theta}_N$.
  end
end

Algorithm 2: AL with SE + FT

Input : Input for Algorithm 1 + regularization parameter λ.
Ouput: A classifier $f(\cdot; \boldsymbol{\theta}_*)$.
Randomly draw m samples from $\mathcal{U}$, remove them from $\mathcal{U}$, and set them as $\mathcal{D}_{\text{train}}$.
Randomly initialize $\boldsymbol{\theta}_0$.
for $t = 1, \ldots, T$ **do**
  `// Only at the final episode!`
  if $t = T$ **then**
    Randomly initialize $\boldsymbol{\theta}_0$.
  end
  Set $\boldsymbol{\Theta} \leftarrow \varnothing$.
  for $j = 1, \ldots, N$ **do**
    Draw a mini-batch $\mathcal{B}$ from $\mathcal{D}_{\text{train}}$.
    $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_{j-1} - \eta(j) \nabla_{\boldsymbol{\theta}} (\mathcal{J}(\mathcal{B}, \boldsymbol{\theta}_{j-1}) + \lambda \mathbb{1}_{\{t>1\}} \|\boldsymbol{\theta}_{j-1} - \boldsymbol{\theta}_0\|^2)$.
    if $j \geq N_{\text{thres}} \wedge \mathrm{mod}\,(j - N_{\text{thres}}, \lfloor \frac{N - N_{\text{thres}}}{S} \rfloor) = 0$ **then**
      $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} \cup \{\boldsymbol{\theta}_j\}$.
    end
  end
  if $t < T$ **then**
    Compute $a(\boldsymbol{x}, \boldsymbol{\Theta})$ for all $\boldsymbol{x} \in \mathcal{U}$.
    Pick top m samples, remove them from $\mathcal{U}$, and append them to $\mathcal{D}_{\text{train}}$
    `// Reuse in the next episode.`
    $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}_N$.
  else
    Set $\boldsymbol{\theta}_* \leftarrow \theta_N$.
  end
end

(McCloskey and Cohen, 1989), so the acquisition function based on it may be biased towards recently acquired data. In order to prevent this, we adopt the idea of using replay buffer (Jung et al., 2016; Rolnick et al., 2019; Aljundi et al., 2019), where we draw some portion of the data from the newly acquired data and the remaining portion from the past data. We empirically find that this significantly improves the stability of the acquisition functions.

Regularization. Similar to the replay buffer, we regularize the fine-tuning procedure not to deviate too much from the previous parameter. That is, optimize the parameter $\boldsymbol{\theta}$ with the ℓ_2-regularizer $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2$ where $\boldsymbol{\theta}_0$ is the starting point of the fine-tuning (the parameters passed from the previous episodes). We also find this regularization improves the quality of the acquisition functions, leading to better classification accuracy in the final episode.

Algorithm 2 summarizes the AL with fine-tuning. The parts that are different from the AL without fine-tuning is marked as blue.

4 Related Works

Active Learning. Based on how an unlabeled example is fed into an AL agent, AL can broadly be categorized into membership query synthesis, where the agent even generates examples in the sample space, stream-based selective sampling, where the agent decides whether a given input is helpful if labeled in online settings, and pool-based active learning, where the agent can access a large unlabeled pool (Settles, 2009). Pool-based active learning can be further divided into uncertainty-based approach, diversity-based approach, and hybird approach (Ren et al., 2021). Geifman and El-Yaniv (2019) first introduced neural architecture search into AL, claiming that the over-parameterized model could lead to overfitting especially in the earlier episodes with fewer labeled examples, and therefore, uncertainty estimates could be inaccurate. Similarly, Munjal et al. (2022) argued that the

optimal hyperparameters may vary when the training set size increases and used Bayesian hyperparameter optimization (AutoML) every episode.

Ensemble. Ensembles of neural network have shown competitive performance improvement and are widely used in machine learning and deep learning. In addition, it shows improvements in estimates for predictive uncertainty. DE (Lakshminarayanan et al., 2017) is one of the well-behaved methods estimating the uncertainty of deep neural networks. This method works with several classifiers trained with the same dataset and architecture but with different seeds for a random number generator. However, the biggest limitation of ensemble is its computational costs because it requires training multiple models (Izmailov et al., 2018).

Active Learning with Ensemble. The uncertainty of an example for neural networks can be estimated by using an ensemble of neural networks in the context of AL. Gal et al. (2017) used BALD (Houlsby et al., 2011) on MCDO (Gal and Ghahramani, 2017) and later extended it to batch settings where it considers overlaps among acquired data points (Kirsch et al., 2019). However, due to the expensive cost of training multiple models, most research on ensemble-based AL had been restricted to the traditional ML algorithms (Melville and Mooney, 2004; Körner and Wrobel, 2006). Beluch et al. (2018) compares the performance of various acquisition functions and uncertainty estimation methods on large-scale image classification tasks, proving that DE consistently outperforms other uncertainty-based methods such as MCDO or a single model (Gal et al., 2017). In the same manner, Bayesian neural networks shows much robustness and reliability than MCDO in the context of AL with continual learning (Rakesh and Jain, 2021). However, many previous works have limited their experiments on small-scale datasets using models with less capacity or on datasets with many redundant examples and sometimes reported the performance of their acquisition strategies without separating the acquisition quality from the classifier training method.

5 Experiments

In this section, through an extensive empirical comparison on three image classification benchmarks (CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and Tiny ImageNet (Le and Yang, 2015)), we would like to demonstrate the followings.

- Measuring epistemic uncertainty via SE for AL is effective, comparable or even better than the one based on DE, across various choices of uncertainty-based acquisition functions.
- AL with SE + FT can significantly reduce the training cost without sacrificing too much accuracy.
- The reason why SE is effective for AL is that it can build a committee of models yielding diverse predictions. Interestingly, for the purpose of acquiring samples of better qualities, the diversity in predictions of the models is more important than the accuracy of the models.

We compare the AL algorithms with three variables - acquisition functions (VR, ME, BALD), algorithm to measure uncertainty (DE, SE, MCDO), and how to train the classifier in the final episode (single model via a vanilla SGD, DE). We report the results with ResNet-18 (He et al., 2016). For more details such as experimental protocols or hyperparameter settings, please refer to Appendix A. The results on CIFAR-10, CIFAR-100, and Tiny ImageNet are summarized in Table 1, Table 2, and Table 3, respectively. Due to the limited resource, we report the results with three acquisition functions for CIFAR-10, and report the results with only VR for CIFAR-100 and Tiny ImageNet.

5.1 Analysis of the main results

Effectiveness of SE for AL. Table 1 confirms that the measuring uncertainty with SE acquires the samples leading to the best classification accuracies, regardless of the choice of acquisition functions or how we train the final classifier. This is somewhat remarkable, considering that the runtimes of SE based methods are not significantly longer than those for the baselines, while DE requires much longer runtimes due to its requirement to train and test with multiple models. It is also notable that the choice of uncertainty estimation method was much more important than the choice of acquisition functions. A similar trend is observed for CIFAR-100 and Tiny ImageNet; overall, SE leads to the

Table 1: Results on CIFAR-10

Acq Fn	Uncertainty	A single model at the final episode				DE at the final episode				Runtime
		10%	15%	20%	25%	10%	15%	20%	25%	
VR	SE	69.10±0.46	79.60±0.55	84.22±0.49	87.37±0.26	75.01	83.38	87.22	89.75	15.3 hr
	DE	66.43±0.75	77.44±0.73	82.48±0.83	86.34±0.44	73.23	81.77	85.96	88.95	60.5 hr
	MCDO	67.73±0.76	76.98±0.45	81.87±0.52	86.05±0.20	73.12	81.52	85.70	89.10	13.3 hr
	SE + FT	69.72±0.57	79.36±0.33	84.24±0.23	87.06±0.28	74.94	83.11	87.14	89.26	2.3 hr
BALD	SE	70.80±0.06	79.63±0.37	84.32±0.35	87.43±0.27	76.37	83.19	86.86	89.25	15.3 hr
	DE	68.36±1.16	78.10±0.56	82.43±0.66	86.12±0.30	74.07	82.23	85.67	88.81	60.5 hr
	MCDO	69.28±0.48	77.32±0.24	82.85±0.27	86.15±0.28	74.03	81.17	86.41	88.52	13.3 hr
ME	SE	68.39±0.84	79.11±0.58	84.19±0.20	87.37±0.26	74.79	82.99	86.82	89.65	15.3 hr
	DE	65.77±1.39	77.31±0.85	82.54±0.26	86.24±0.29	72.30	82.08	85.97	88.97	60.5 hr
Random		69.88±0.25	78.08±0.37	82.44±0.38	84.58±0.089	74.85	81.32	84.95	86.92	12.1 hr

Table 2: Results on CIFAR-100

Acq Fn	Uncertainty	A single model at the final episode				DE at the final episode				Runtime
		16%	18%	20%	22%	16%	18%	20%	22%	
VR	SE	59.34±0.36	61.02±0.11	63.02±0.23	64.14±0.27	64.26	66.21	68.27	69.24	7.2 hr
	DE	58.17±0.20	60.44±0.21	62.08±0.29	63.46±0.35	63.78	65.99	67.99	69.22	28.7 hr
	MCDO	58.81±0.34	60.62±0.26	62.38±0.26	63.50±0.17	63.23	65.09	66.80	67.69	6.2 hr
	SE + FT	59.56±0.58	61.17±0.37	62.88±0.26	64.81±0.27	64.35	66.23	67.68	69.72	0.8 hr
Random		58.93±0.20	60.58±0.29	61.95±0.24	62.80±0.44	63.63	65.09	66.68	67.82	5.7 hr

best classification accuracies. We set $S = 5$ for SE and DE and $S = 25$ for MCDO. We repeated the experiment five times, and the reported means and standard deviations are averaged over 5 trials and 1 trial for retraining a single model and DE, respectively.

Effectiveness of fine-tuning. In Tables 1 to 3, we compare the AL with SE + FT (Algorithm 2) to the baselines with VR as acquisition functions. Somewhat surprisingly, SE + FT achieved comparable or even better test accuracies given much shorter runtimes. Section 5.2 shows the progression of test accuracies during the episodes of SE + FT procedure. Before the final episode, the intermediate models (used only for the acquisitions) exhibit lower accuracies. This is somewhat expected as they are fine-tuned and trained for less number of epochs. At the final episodes where a new classifier is trained from scratch, the test accuracy catches up with the ordinary AL, indicating that even if the accuracies of the intermediate classifiers are inferior, the samples acquired with them are good enough to obtain the decent classifier at the final episode. We set $S = 5$ for SE + FT, and the reported means and standard deviations are averaged over 5 trials and 1 trial for retraining a single model and DE, respectively.

AL with pretrained models. As one can see from Table 3, the performances of the methods on Tiny ImageNet is generally bad, presumably due to the complexity of the dataset. To this end, we employed models pretrained from ImageNet (Deng et al., 2009), and tested whether the AL algorithms could fine-tune on pre-trained models. We tested ResNet-50 and vision transformer (Dosovitskiy et al., 2021) backbones, and compared SE-based and DE-based AL with VR acquisition. For both backbones, SE significantly outperformed DE, especially with the small number of acquisitions. Here, we set $S = 5$ for SE and $S = 3$ for DE, and the reported means and standard deviations are averaged over 3 trials and 1 trial for retraining a single model and DE, respectively.

5.2 Analysis of the uncertainty estimation.

In this section, we analyze SE more in detail to see why it is more effective than DE for AL. We conjecture that SE builds a committee of models whose predictions are more diverse than the ones from other methods, and this is a key to its success in AL. To verify this, We measure average KL-divergences and pair-wise disagreement values (Melville and Mooney, 2005) of the predictions computed from SE, DE, and MCDO. The disagreement bewteen two class probabilities $f(\cdot; \boldsymbol{\theta}^{(i)})$ and

Table 3: Results on Tiny ImageNet

Acq Fn	Uncertainty	A single model at the final episode				DE at the final episode				
ResNet-18 from scratch		14%	15%	16%	17%	14%	15%	16%	17%	Runtime
VR	SE	30.60±0.32	31.28±0.16	32.38±0.47	33.37±0.24	36.23	37.49	38.58	39.79	14.0 hr
	DE	30.05±0.21	31.20±0.43	31.63±0.28	32.81±0.65	35.83	37.17	37.97	39.34	26.1 hr
	MCDO	29.48±0.30	31.40±0.89	32.35±0.33	33.32±0.68	35.12	37.33	38.16	38.94	14.8 hr
	SE + FT	30.86±0.41	31.77±0.39	32.96±0.26	33.74±0.31	36.32	37.13	38.19	39.01	2.3 hr
Random		28.37±0.45	28.99±0.44	29.04±0.12	29.19±0.39	32.17	33.23	32.94	32.90	8.5 hr
ResNet-50 pretrained		3%	5%	7%	10%	3%	5%	7%	10%	Runtime
VR	SE	60.89±0.40	66.46±0.30	69.71±0.42	72.31±0.14	62.72	68.59	71.01	73.85	3.2 hr
	DE	59.88±0.21	61.81±0.43	68.21±0.28	70.44±0.65	62.19	67.14	70.18	73.23	5.8 hr
Random		61.82±0.34	64.11±0.22	68.76±0.07	70.75±0.16	63.40	67.59	69.64	72.09	1.8 hr
ViT-Base pretrained		3%	5%	7%	10%	3%	5%	7%	10%	Runtime
VR	SE	77.12±1.10	80.68±2.78	85.31±0.74	86.39±0.14	80.51	85.81	86.95	88.73	7.1 hr
	DE	69.03±5.47	73.20±5.24	82.83±1.32	86.17±0.26	77.01	75.92	85.86	88.06	14.9 hr
Random		71.43±1.03	75.98±2.59	82.20±0.57	84.74±0.92	78.73	84.50	86.83	86.87	4.9 hr

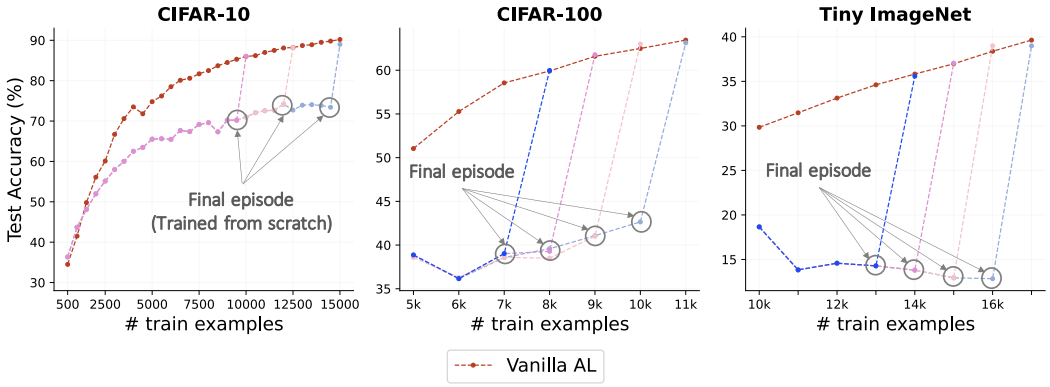

Figure 1: Results of vanilla AL (red) and fine-tuning (FT) at specific episodes (the rest of the colors). CIFAR-10, CIFAR-100 and Tiny ImageNet all used ResNet-18 architecture. Note that before the final episode, the intermediate model show low accuracies.

$f(\cdot;\boldsymbol{\theta}^{(j)})$ on an example $\boldsymbol{x}$ is calculated as

$$d_{i,j} = \mathbb{1}_{\{\arg\max_k f_k(\boldsymbol{x};\boldsymbol{\theta}^{(i)}) \neq \arg\max_k f_k(\boldsymbol{x};\boldsymbol{\theta}^{(j)})\}}. \tag{4}$$

As summarized in Fig. 2 (left), in general, SE exhibits much higher KL-divergences and disagreement values between among its predictions compared to DE and MCDO. As reported in the literature (Fort et al., 2021), DE shows higher disagreements than MCDO, but still much less than SE. We believe that this is mainly due to the nature of active learning where we usually work with relatively small amount of data and less number of training epochs than typical supervised learning settings; While DE parameters are collected at the end of each training runs, so the models are likely to reach the local optimum. On the other hand, SE collects the parameter snapshots during a single training run, so some of them may not converge to the local optimum. This degrades the classification accuracy, but as we have pointed out several times, for the purpose of acquisition, the diversity in predictions are more important then the accuracy of the models. Fig. 2 (right) shows the correlation between the VR values and the class prediction probabilities, along with the histogram of VR values among the acquired samples. SE finds more samples having larger VR values that actually are predicted to be confusing samples for the classifier.

As a qualitative comparison, in Fig. 3, we show the map of class prediction probabilities for the committee constructed from SE. For the samples having low VR values, the members of the committee tend to agree with each other, but for the ones with high VR values, the members disagree with each other as expected.

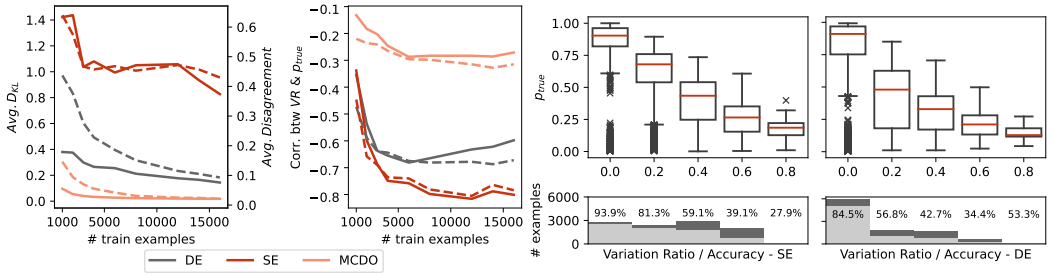

Figure 2: (Left) average KL-Divergence ▬ and disagreement ▭ between the predictions with different uncertainty estimation methods. (Right) Spearman's rank correlations ▬ and Pearson's correlations ▭ between VR and a predicted probability of the actual class prediction probabilities. In the bottom of (right) pannel, we display histograms of samples binned with the VR values.

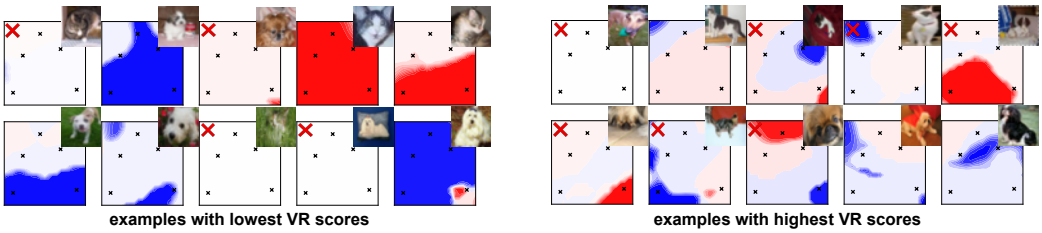

Figure 3: Probability maps for the images given in the upper right corners where the blue means class dog and red means class cat. Black crosses denote the snapshots obtained from SE in the parameter space, and red crosses on the upper left corners indicates that a committee of models predicted it wrongly.

6 Conclusion

In this paper, we demonstrate that estimating epistemic uncertainties of the predictions using SE works efficiently for uncertainty-based AL. Through an extensive experiments with real-world image classification benchmark, we empirically confirmed that the AL with SE outperforms AL with DE or MCDO, for various choices of acquisition functions. We further presented a novel AL algorithm based on fine-tuning, where we keep a single model and continuously fine-tune it instead of re-initializing the models at the beginning of every episode. The resulting algorithm could achieve comparable classification accuracies given the same number of samples acquired compared to the baselines with a much less number of training steps. We provide a detailed analysis of the effectiveness of SE for AL and show that SE builds model committees that yield diverse predictions that are useful for acquiring informative samples.

7 Reproducibility Statement

We used the Pytorch (Paszke et al., 2019) library in our experiments and our algorithms which are described in Algorithm 1 and Algorithm 2. In addition, all experimental details and configurations on hyperparameters are recorded in Appendix A. We will provide an open-source implementation of AL environments and our code of SE and SE + FT algorithm.

References

R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019. 5

A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*

(ICLR), 2020. 2

W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9368–9377, 2018. 2, 6, 12

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112:859–877, 2017. 2

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *Proceedings of The 32nd International Conference on Machine Learning (ICML 2015)*, 2015. 3

F. D'Angelo and V. Fortuin. Replulsive deep ensembles are Bayesian. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 2021. 4

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, 2009. 7

a. Dosovitskiy, L. Beyer, A. Kolsenikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 7

S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: a loss landscape perspective. *arXiv preprint arXiv:2106.11642*, 2021. 2, 4, 8

L. C. Freeman. *Elementary applied statistics: for students in behavioral science*. John Wiley and Sons, 1965. 3

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning (ICML 2016)*, 2017. 1, 6

Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017. 1, 3, 6

T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, 2018. 2, 4

Y. Geifman and R. El-Yaniv. Deep active learning with a neural architecture search. *Advances in Neural Information Processing Systems*, 32, 2019. 5

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

N. Houlsby, F. Huzár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011. 1, 3, 6

G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: train 1, get M for free. In *International Conference on Learning Representations (ICLR)*, 2017. 2, 4

P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 6

H. Jung, J. Ju, M. Jung, and J. Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016. 5

A. Kirsch, J. Van Amersfoort, and Y. Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32, 2019. 6

C. Körner and S. Wrobel. Multi-class ensemble-based active learning. In *European conference on machine learning*, pages 687–694. Springer, 2006. 6

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017. 2, 4, 6

Y. Le and X. Yang. Tiny imagenet visual recognition challenge. Technical report, Stanford University, 2015. 6

C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017. 2

M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 5

P. Melville and R. J. Mooney. Diverse ensembles for active learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 74, 2004. 6

P. Melville and R. J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005. 7

P. Munjal, N. Hayat, M. Hayat, J. Sourati, and S. Khan. Towards robust and reproducible active learning using neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 223–232, 2022. 5, 12

Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. 2

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 9, 12

V. Rakesh and S. Jain. Efficacy of bayesian neural networks in active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2601–2609, 2021. 2, 6

P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021. 1, 5

D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 5

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. 5

L. N. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019. 12

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014. 3

R. Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 12

A. G. Wilson and P. Izmailov. Deep ensembles as approximate Bayesian inference. https://cims.nyu.edu/~andrewgw/deepensembles/, 2021. 4

D. Yoo and I. S. Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 93–102, 2019. 12

A Experimental Details

A.1 Snapshot Ensemble vs. Deep Ensemble

A.1.1 Baselines

When we designed the experiments, one of the first criteria was the robustness, reproducibility, and generalizability. For CIFAR-10 and CIFAR-100, we scored 90.15% and 68.20% test accuracy (averaged over five trials), respectively, with VR acquisition function until acquiring 30% of the labels with the settings mentioned above, while Munjal et al. (2022) reported a baseline with 90.87% and 59.36% test accuracy with 40% random samples. For the first experiment in every experiment not using , We randomly draw samples

We will not only release code and configurations, but also disclose the indices of examples queried by acquisition functions and strategies on each dataset. Reported runtimes are based on an Ubuntu 20.04 server with AMD Ryzen-4 5900X CPU with 64GB RAM, NVIDIA RTX-3090 GPU with 24GB VRAM. We additionally used FP16 (Automatic Mixed Precision) for Tiny-ImageNet for faster training.

A.1.2 Hyperparameters

We used a mini-batch of size 64 and a standard SGD optimizer with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.01 for both CIFAR-10 and CIFAR-100 dataset. During SE, a learning rate was raised to 0.01 and 0.005 for CIFAR-10 and CIFAR-100, respectively. For faster convergence and reducing efforts in finding optimal hyperparameters, One Cycle learning rate scheduler (ONECYCLE) proposed by Smith and Topin (2019) with max_lr set to 0.01 has been adopted for CIFAR-10 and CIFAR-100 datasets. For augmentations, we normalized images with the mean and the variance of all images in the train set and applied random horizontal flip for both datasets. For CIFAR-100, random cropping was also applied. The total number of training epochs is 200, and for SE we collected 5 snapshots with additional 50 epochs (10 epoch interval).

Without ONECYCLE, the performance dropped from 84.1% to 81.2% for Random, 86.0% (Δ 1.9%p) to 84.1% (Δ 2.9%p) for VR, 85.4% (Δ 1.3%p) to 83.4% (Δ 2.2%p) for BALD, 85.2% (Δ 1.1%p) to 84.1% (Δ 2.9%p) for ME at episode 20 (10,000 labeled examples) on CIFAR-10 with ResNet-18, respectively (averaged over 5 trials, in parentheses are the differences between random sampling and the acquisition function). Although the differences are almost twice as ones without ONECYCLE, we decided to report the figures with the higher performance in that it is more suitable to real-world applications.

For Tiny-ImageNet dataset with ResNet-18, We also used a SGD optimizer with momentum of 0.9 and weight decay of 0.0001. We adjusted the learning rate to 0.1 for the first half of the training epochs, 0.01 until the 75% of the training epochs, and 0.001 for the rest. During SE, the learning rate is increased to 0.05. For augmentations, we used random crops and random horizontal flips, which are the *de-facto* standard random augmentation strategies when reporting in academia. To reduce the computational overhead in ensemble-based acquisitions due to multiple forward passes of a large unlabeled pool $\mathcal{U}$, we first randomly draw Q unlabeled examples from the pool and measured the scores, following Beluch et al. (2018); Yoo and Kweon (2019). We set Q to 10,000. The total number of training epochs is 100, and for SE we collected 5 snapshots with additional 50 epochs (10 epoch interval).

For transfer learning experiments on Tiny ImageNet, we used pretrained weights for ResNet-50 from torchvision (Paszke et al., 2019) and those for ViT-base-224 from timm (Wightman, 2019) and replaced the final linear classification head. Instead of using the original image size, an image was scaled up to 224×224 resolution to fit the model trained with ImageNet dataset. The total number of training epochs is 50, and for SE we collected 5 snapshots with additional 25 epochs (5 epoch interval). We here also used a SGD optimizer with momentum of 0.9 and weight decay of 0.0001. For ResNet-50, we used ONECYCLE (the same as above) and for ViT-base, we did not use a learning rate scheduler (constant learning rate). We also set Q to 10,000. Note that Table 4 for summary of hyperparameters.

Table 4: Hyperparameters

Dataset	architecture	optimizer	learning rate	momentum	weight decay	scheduler	SE learning rate	max epoch	SE epoch	# snapshot
CIFAR-10	ResNet-18	SGD	0.001	0.9	0.01	ONECYCLE	0.01	200	50	5
	VGG-16	SGD	0.001	0.9	0.01	ONECYCLE	0.01	100	100	5
CIFAR-100	ResNet-18	SGD	0.001	0.9	0.005	ONECYCLE	0.01	200	50	5
Tiny ImageNet	ResNet-18	SGD	0.001	0.9	0.0001	Steps	0.01	100	50	5
	ResNet-50	SGD	0.001	0.9	0.0001	ONECYCLE	0.05	50	25	3
	ViT-base-224	SGD	0.001	0.9	0.0001		0.05	50	25	3

Table 5: Test accuracy at episode 30 according to the number of training data on CIFAR-10

# train data	500	1000	1500	2000	2500	3000
Acc (%)	87.79	88.26	88.57	89.30	89.39	88.91

A.1.3 MODEL STRUCTURES

For all experiments above, the structure of ResNet-18 model is slightly modified in order to fit 32×32 and 64×64 images following the standard protocol as follows:

- The kernel size of the first convolution layer (conv1) has changed to 3, and the stride is changed to 1.
- The max pooling layer is disabled.
- For MCDO, a dropout layer with dropout rate $p = 0.5$ is added in front of the final linear classifier layer, since the original implementation has no dropout layers.

Similarly, the structure of VGG-16 model is slightly modified as follows:

- The average pooling layer is disabled.
- No additional dropout layer is attached to the model since its classifier already has two dropout layers with dropout rate $p = 0.5$. We turned on the dropout layer when querying.

Adding MCDO layers has no effects on reported figures at Section 5 since we reported acquisition qualities by retraining the model with the queried examples.

A.2 ACQUISITION QUALITY

For fair comparison, we designed an experiment to measure the acquisition quality. The data selected by SE, DE, and MCDO were retrained in the single model and the ensemble model, respectively, to compare how effective the selected data showed in learning. The experimental details are the same as Appendix A.1.

A.3 FINE TUNING

We used the same experimental details as Section A.1 except for the maximum number of epoch and the number of data used for training. CIFAR-10, CIFAR-100, and Tiny ImageNet all used up to 50 epochs except for the last episode and took 5 snapshots start from 40 epoch.

A.3.1 NUMBER OF RANDOM SAMPLED DATA

In the case of the number of data used for fine tuning process, some of the data labeled in the previous episodes were added to the newly acquired data. In case of CIFAR-10, we used budget size for 500. Moreover, we add additional 2,000 randomly sampled data from previous labeled data. CIFAR-100 and Tiny ImageNet, we add 1,000 random sampled from labeled data to 1,000 newly acquired data. In Table 5, there are the test accuracy according to the number of training data when it trained from scratch at episode 30.

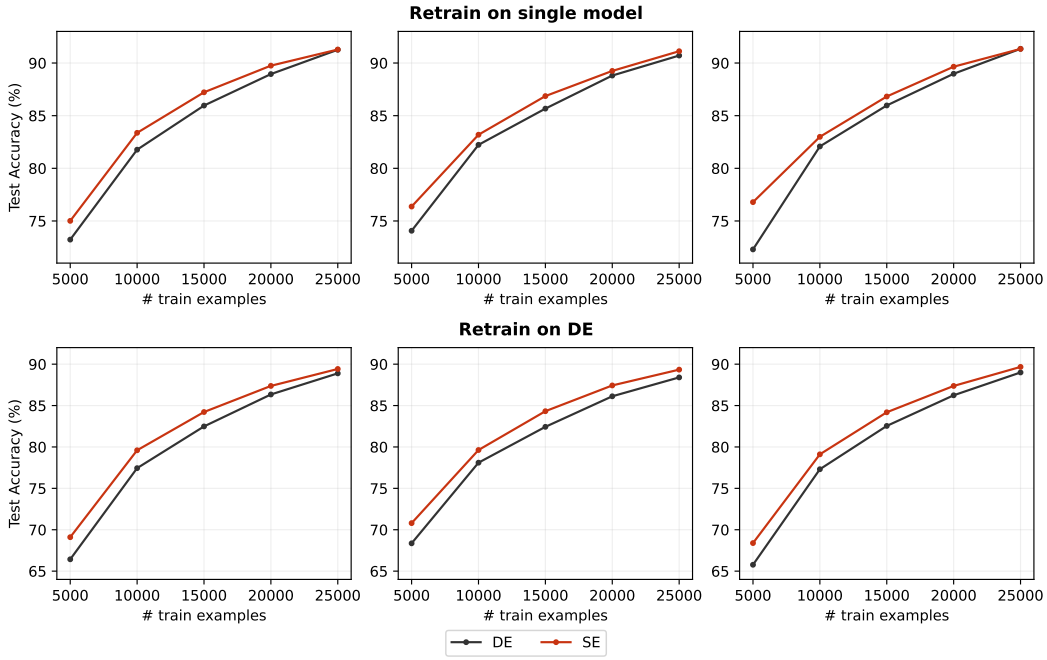

Figure 4: Acquisition quality of VR (left), BALD (middle), ME (right) on CIFAR-10

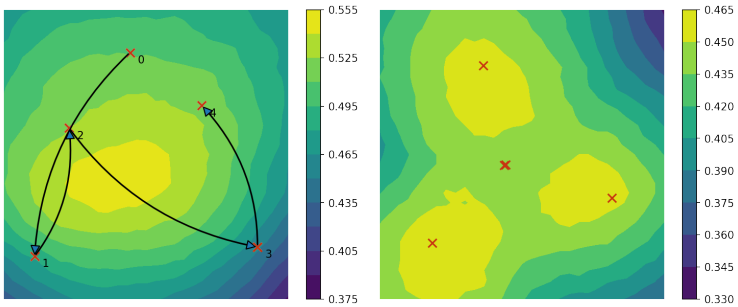

Figure 5: Loss surface on test set of SE (left) and DE with same initialization for the visualization purpose (right) in the parameter space when trained with the first 2,000 examples of CIFAR10 dataset. Contours represent test accuracy, and red points denote weights gathered for AL. With a high learning rate, SE picks checkpoints that are weak individually but strong together around the wider optimum compared to DE, which falls into their own narrow optima.

A.3.2 REGULARIZATION HYPERPARAMETER

For regularization, it needs to find λ of ℓ_2-regularizer $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2$. In Table 6, there are the fine tuning results depending on λ value. Note that, all the results shown in the table mean the last episode performance.

Table 6: Test accuracy according to the regularization hyperparameter

λ	0	0.001	0.005	0.01	0.02
Episode 20	85.58	88.66	86.46	85.79	86.05
Episode 25	88.30	87.83	87.73	87.63	88.19
Episode 30	88.30	88.59	88.90	89.39	89.00

Table 7: Comparison between SE and FT with VGG-16 on CIFAR-10

	Retrain on a single model				Retrain on DE			
	10%	15%	20%	25%	10%	15%	20%	25%
SE	59.79±2.36	68.18±2.01	68.88±1.89	72.16±1.21	66.97	74.65	77.60	81.10
SE + FT	57.78±3.31	65.22±2.21	68.58±0.80	72.50±1.20	68.43	74.84	78.20	81.45

A.3.3 VGG-16

We also used VGG-16 architecture for fine tuning experiments in Table 7. Note that, compared to SE's performance, FT has caught up and even shows better performance.