Towards Robust and Reproducible Active Learning using Neural Networks

Prateek Munjal [*] Nasir Hayat [†] Munawar Hayat [‡] Jamshid Sourati [§] Shadab Khan [*]

Abstract

Active learning (AL) is a promising ML paradigm that has the potential to parse through large unlabeled data and help reduce annotation cost in domains where labeling data can be prohibitive. Recently proposed neural network based AL methods use different heuristics to accomplish this goal. In this study, we demonstrate that under identical experimental settings, different types of AL algorithms (uncertainty based, diversity based, and committee based) produce an inconsistent gain over random sampling baseline. Through a variety of experiments, controlling for sources of stochasticity, we show that variance in performance metrics achieved by AL algorithms can lead to results that are not consistent with the previously reported results. We also found that under strong regularization, AL methods show marginal or no advantage over the random sampling baseline under a variety of experimental conditions. Finally, we conclude with a set of recommendations on how to assess the results using a new AL algorithm to ensure results are reproducible and robust under changes in experimental conditions. We share our codes to facilitate AL evaluations. We believe our findings and recommendations will help advance reproducible research in AL using neural networks.

Abbreviations: Active Learning (AL), Random Sampling Baseline (RSB), Query-by-Committee (QBC), Variational Adversarial Active Learning (VAAL), Uncertainty-based sampling (UC), Deep Bayesian Active Learning (DBAL), Bayesian Active Learning by Disagreement (BALD), Random Augmentation (RA), Stochastic Weighted Averaging (SWA), Shake-Shake regularization (SS).

Conventions: Models that are regularized using either one or combination of RA, SWA, and SS have been identified with a suffix *-SR* added to the abbreviation, SR signifies 'strong regularization'. Models without such a suffix are also regularized, but with standard methods such as weight decay and data augmentation using random flip and horizontal crop.

[*]G42 Healthcare, Abu Dhabi, UAE
[†]NYUAD, UAE
[‡]Monash University, Australia
[§]University of Chicago, Chicago, IL, USA
Correspondence to: Shadab Khan <skhan.shadab@gmail.com>

1. Introduction

Active learning (AL) is a machine learning paradigm that promises to help reduce the burden of data annotation by intelligently selecting a subset of informative samples from a large pool of unlabeled data that. In AL, a model trained with a small amount of labeled seed data is used to parse through the unlabeled data to select the subset that should be sent to an oracle (annotator). To select such a subset, AL methods rely on exploiting the learned latent-space, model uncertainty, or other heuristics. The promise of reducing annotation cost has brought a surge of interest in AL research [2, 3, 7, 14, 16, 25, 27, 28, 32] and with it, a few outstanding issues. **First**, The results reported for Random sampling baseline, RSB vary significantly between studies. For example, using 20% labeled data of CIFAR10, the difference between RSB performance reported by [32] and [28] is $\approx 13\%$ under identical settings. **Second**, The results reported for the same AL method can vary across studies: using VGG16 [26] on CIFAR100 [17] with 40% labeled data, Coreset [25] reported $\approx 55\%$ classification accuracy whereas VAAL [27] reported 47.01% using the method reported in [25]. **Third**, Recent AL studies have been inconsistent with each other. For example, [25] and [6] state that diversity-based AL methods consistently outperform uncertainty-based methods, which were found to be worse than the RSB. In contrast, recent developments in uncertainty based studies [32] suggest otherwise.

In addition to these issues, results using a new AL method are often reported on simplistic experimental conditions - (i) regularization is not sufficiently explored beyond the usual methods (e.g. weight decay), (ii) with increasing AL iterations, the training data distribution changes, however, the training hyper-parameters are fixed in advance. Such issues in the AL results has spurred a recent interest in benchmarking of AL methods and recent NLP and computer vision studies have raised a number of interesting questions [20, 22, 24]. With the goal of improving the reproducibility and robustness of AL methods, in this study we evaluate the performance of these methods for image classification task.

Contributions: Through a comprehensive set of experiments performed using consistent settings under a common code base (PyTorch-based[1]) we compare differ-

[1]https://github.com/PrateekMunjal/TorchAL

ent AL methods including state-of-the-art diversity-based, uncertainty-based, and committee-based methods [2, 7, 25, 27] and a well-tuned RSB. We demonstrate that: **1)** With strong regularization and hyper-parameters tuned using AutoML, RSB performs comparably to AL methods in contrast to the previously reported results in the literature. **2)** No AL method consistently outperforms other approaches, and conclusions can change with different experimental settings (*e.g.* using a different architecture for the classifier or with different number of AL iterations). **3)** The difference in performance between the AL methods and the RSB is much smaller than reported in the literature. **4)** With a strongly-regularized model, the variance in accuracy achieved using AL methods is substantially lower across consistent repeated training runs, suggesting that such a training regime is unlikely to effect misleading results in AL experiments. **5)** Finally, we provide a set of guidelines on experimental evaluation of a new AL method.

2. Pool Based Active Learning Methods

Contemporary pool-based AL methods can be broadly classified into: **(i)** uncertainty based [7, 16, 27], **(ii)** diversity based [6, 25], and **(iii)** committee based [2]. AL methods also differ in other aspects, for example, some AL methods use the task model (e.g. model trained for image classification) within their sampling function [7, 25], where as others use different models for task and sampling functions [2, 27]. These methods are discussed in detail next.

Notations: Starting with an initial set of labeled data L_0^0=$\{(x_i, y_i)\}_{i=1}^{N_L}$ and a large pool of unlabeled data U_0^0=$\{x_i\}_{i=1}^{N_U}$, pool-based AL methods train a model Φ_0. A sampling function $\Psi(L_0^0, U_0^0, \Phi_0)$ then evaluates $x_i \in U_0$, and selects k (budget size) **samples** to be labeled by an oracle. The selected samples with **oracle-annotated** labels are then added to L_0^0, resulting in an extended L_0^1 labeled set, which is then used to **retrain** Φ. This cycle of **sample-annotate-train** is repeated until the sampling budget is exhausted or a satisficing metric is achieved. AL sampling functions evaluated in this study are outlined next.

2.1. Model Uncertainty on Output (UC)

[19] ranks the unlabeled data, $x_i \in U$ in a descending order based on their scores given by $\max_j \Phi(x_i); j \in \{1 \dots C\}$ where C is the number of classes, and chose the top k samples. Typically this approach focuses on the samples in U for which the softmax classifier is least confident. Recently, Huang et al. [11] proposed to measure the uncertainty by measuring the output discrepancies from the model trained at different AL cycles.

2.2. Deep Bayesian Active Learning (DBAL)

Gal et al. [7] train the model Φ with dropout layers and use monte carlo dropout to approximate the sampling from posterior. For our experiments, we used the two most reported acquisitions *i.e.* max entropy and Bayesian Active Learning by Disagreement (BALD). The max entropy method selects the top k data points having maximum entropy as $\arg\max_i \mathbb{H}[P(\mathbf{y}|x_i)]; \forall x_i \in U_0$ where the posterior is given by, $P(\mathbf{y}|x_i) = \sum_{j=1}^{T} \frac{1}{T} P(\mathbf{y}|x_i, \phi_j)$. Here T denotes number of forward passes through the model, Φ. BALD selects the top k samples that increase the information gain over the model parameters *i.e.* $\arg\max_i \mathbb{I}[P(\mathbf{y}, \Phi|x_i, L_0)]; \forall x_i \in U_0$ We implement DBAL as described in [7] where probability terms in information gain is evaluated using previous equation.

2.3. Coreset

Sener et al. [25] exploit the geometry of data points and choose samples that provide a cover to all data points. Essentially, their algorithm tries to find a set of points (cover-points), such that distance of any data point from its nearest cover-point is minimized. They proposed two sub-optimal but efficient solutions to this NP-Hard problem: coreset-greedy and coreset-MIP (Mixed Integer programming), coreset-greedy is used to initialize coreset-MIP. For our experiments, following [32], we implement coreset-greedy since it achieves comparable performance while being significantly compute efficient.

2.4. Variational Adversarial Active Learning

Sinha et al. [27] combined a VAE [15] and a discriminator [9] to learn a metric for AL sampling. VAE encoder is trained on both L and U, and the discriminator is trained on the latent space representations of L and U to distinguish between seen (L) and unseen (U) images. Sampling function selects samples from U with lowest discriminator confidence (to be seen) as measured by output of discriminator's softmax. Effectively, samples that are most likely to be unseen based on the discriminator's output are chosen. Since VAAL does not account for the end task, recent methods such as SRAAL [35], TAVAAL [14] have incorporated the task awareness too.

2.5. Ensemble Variance Ratio Learning

Proposed by [2], this is a query-by-committee (QBC) method that uses a variance ratio computed as $v = 1 - f_m/N$. This variance ratio select the sample set with the largest dispersion (v), where N is the number of committee members (CNNs), and f_m is the number of predictions in the modal class category. Variance ratio lies in 0–1 range and can be treated as an uncertainty measure. We note that it is possible to formulate several AL strategies using the ensemble e.g. BALD, max-entropy, etc. Variance ratio was chosen for this study because it was shown by authors to lead to superior results. For training the CNN ensembles, we train 5 models with VGG16 architecture but

a different random initialization. Further, following [2], the ensembles are used only for sample set selection, a separate task classifier is trained in fully-supervised manner to do image classification.

3. Regularization and Active Learning

In a ML training pipeline comprising data–model–metric and training tricks, regularization can be introduced in several forms. In neural networks, regularization is commonly applied using parameter norm penalty (metric), dropout (model), or using standard data augmentation techniques such as horizontal flips and random crops (data). However, parameter norm penalty coefficients are not easy to tune and dropout effectively reduces model capacity to reduce the extent of over-fitting on the training data, and requires the drop probability to be tuned. On the other hand, several recent studies have shown promising new ways of regularizing neural networks to achieve impressive gains. While it isn't surprising that these regularization techniques help reduce generalization error, most AL studies have overlooked them. We believe this is because of a reasonable assumption that if an AL method works better than random sampling, then its relative advantage should be maintained when newer regularization techniques and training tricks are used. Since regularization is critical for low-data training regime of AL where the massively-overparameterized model can easily overfit to the limited training data, we investigate the validity of such assumptions by applying regularization techniques to the entire data–model–metric chain of neural network training.

Specifically, we employ parameter norm penalty, random augmentation (RA) [5], stochastic weighted averaging (SWA) [13], and shake-shake (SS) [8]. In RA, a sequence of n randomly chosen image transforms are sequentially applied to the training data, with a randomly chosen distortion magnitude (m) which picks a value between two extremes. For details of extreme values used for each augmentation choice, we refer the reader to work of [4]. SWA is applied on the model by first saving e snapshots of model during the time-course of optimization, and then averaging the snapshots as a post-processing step. The mode of action of the regularization techniques used affect different components of the neural network training pipeline: RA is applied to data, SWA and SS is applied to model, parameter norm penalty affects the metric. In our experiments, the models which are trained using such additional regularization are referred to as strongly-regularized models (SR-models). The hyper-parameters associated with these regularization techniques as well as experiments and their results when applied to neural network training with AL-selected sample sets are discussed in Sec. 6.4.

4. Tuning Hyper-parameters

The performance of deep neural networks is sensitive to the choice of hyper-parameters (e.g. learning rate, optimizer, weight decay, etc.) and there is no deterministic approach to find a combination that yields best results. Most AL methods perform grid search to find a set of hyper-parameters over the initial labeled set, and these hyper-parameters are fixed for the AL iterations [25, 27]. Fixing the hyper-parameters in AL iterations is questionable - with an increase in AL iterations, the size of training data increases, and the distribution changes since AL heuristics are used to draw a new set to be labeled by the oracle. Therefore, the hyper-parameters found to work well in one AL iteration may not work well at further AL iterations. To address this concern, we use AutoML at *each* AL iteration in our implementation, which does 50 trials of random search over the hyper-parameters. To illustrate this point further, in 4 AL iterations for any given AL method, where initial data of 10% is increased to 40% with a budget size of 10%, we train a total of 200 models and choose the best 4 (1 for each AL iteration) to report the performance. This process is repeated for each labeled set partition: L_0^0,L_1^0,L_2^0,L_3^0,L_4^0. To report the variance in accuracy at an AL iteration a labeled partition, say L_0, we re-use the best hyper-parameters founded using L_0^0 and run on L_0^i, where i $\in \{1, 2, 3, 4\}$. Further details regarding the list of hyper-parameters and their range of choices is shared in the supplementary section.

5. Implementation Details

We perform experiments on most commonly used datasets in active learning: CIFAR10, CIFAR100, with limited additional results reported on ImageNet. For details on our training schedule we refer readers to the supplementary. Given a dataset $\mathcal{D}$, we split it into train (T_r), validation (V), and test (T_s) sets. The train set is further divided into the initial labeled (L_0) and unlabeled (U_0) sets. A base classifier $\mathcal{B}$ is first trained, followed by iterations of sample-annotate-train process using various AL methods. Model selection is done by choosing the best performing model on the validation set. For a fair comparison, a consistent set of experimental settings is used across all methods. Hyper-parameters like learning rate (lr) and weight decay (wd) were tuned using AutoML with random search over 50 trials. We make use of the Optuna library [1] to facilitate these experiments. For ImageNet experiments, we relied on previously published hyper-parameters found using AutoML [34]. Models were trained from random initialization in each AL iteration for experiments that used CIFAR 10 and CIFAR 100 datasets. In case of ImageNet, the initial model trained using random drawn seed batch was trained using random initialization, and subsequent models in AL iterations were initialized using final weights from previous

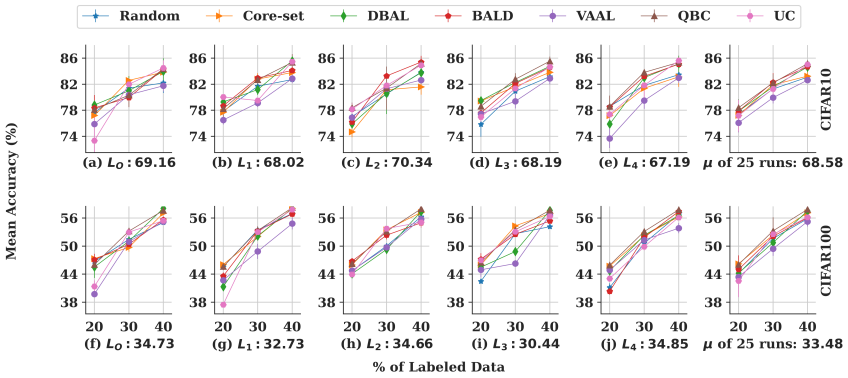

Figure 1. Comparisons of AL methods on CIFAR10 (top) and CIFAR100 (bottom) for different initial labeled sets $L_0, L_1, \cdots, L_4$. The mean accuracy for the base model (at 10% labeled data) is noted at the bottom of each subplot. The model is trained 5 times for different random initialization seeds where for the first seed we use AutoML to tune hyper-parameters and re-use these hyper-parameters for the other 4 seeds. The mean of 25 runs (rightmost column) suggest that no AL method performs consistently better than others.

iteration.

ImageNet Training Details: We use Resnext-50 [29] as our classifier and follow the settings of [34] *i.e.* Optimizer=SGD, $wd = 3 \times 10^{-4}$. We train the base classifier on L_0 for 200 epochs (300 epochs when we include SWA and RA to the training pipeline) where $lr = 0.1$ with a linear warm-up schedule (for first 5 epochs) followed by decaying the lr by a factor of 10 on epoch number: $\{140, 160, 180\}$. For AL iterations we fine-tune the best model (picked by validation set accuracy) from previous iteration for 100 epochs where $lr = 0.01$ which gets decayed by a factor of 10 on epoch number: $\{35, 55, 80\}$. Further, we choose the best model based on a realistically small validation set (*i.e.* 12811 images) following [34]. The input is pre-processed using random crops resized to 224 x 224 followed by horizontal flip (probability=0.5) and normalized to zero mean and one standard deviation using statistics of initial randomly drawn labeled set partition.

Architecture & Hyper-parameters: All experiments are performed using the VGG16 architecture [26] with batchnorm [12], unless otherwise stated. For transferability experiment (refer Sec. 6.5), we use two target architectures *i.e.* 18-layer ResNet [10], and 28-layer 2-head Wide-ResNet (WRN-28-2) [33] in our experiments. Both target architectures are taken from publicly available github repository [18], [21]. For CIFAR10/100 models we set the number of neurons in penultimate fully-connected layer of VGG16 to 512 as in [18]. RA parameteres: N=the number of transformations and M=index of the magnitude, are tuned using AutoML. We empirically select the SWA hyperparameters as: CIFAR 10/100: SWA LR: 5×10^{-4} and frequency: 50. Imagenet: SWA LR: 1×10^{-5} and frequency: 50.

Implementation of AL methods: We developed a PyTorch-based toolkit to evaluate the AL methods in a unified implementation. AL methods can be cast into two categories based on whether or not AL sampling relies on the task model (classifier network). For example, coreset uses the latent space representations learnt by task model to select the sample set, whereas VAAL relies on a separate VAE-discriminator network to select the samples, independent of the task model. In our implementation, we abstract these two approaches in a sampling function that may use the task model if required by the AL method. Each AL method was implemented using a separate sampling function, by referencing author-provided code if it was available. Using command line arguments, the toolkit allows the user to configure various aspects of training such as architecture used for task model, AL method, size of initial labeled set, size of acquisition batch, number of AL iterations, hyper-parameters for task model training and AL sampling and number of repetitions.

Compute: All experiments were performed using 2 available nVidia DGX-1 servers, with each experiment utilizing 1–4 GPUs out of available 8 GPUs on each server. All codes were written in Python using PyTorch and other libraries in addition to third-party code-bases and are made available as part of the supplementary material. Models were trained over a period of many months, AutoML considerably increased time for completing each experiment.

6. Experiments and Results

6.1. Variance in Evaluation Metrics

Training a neural network involves many stochastic components including parameter initialization, data augmentation, mini-batch selection, and batchnorm whose parameters change with mini-batch statistics. These elements can lead to a different optima thus resulting in varying performances across different runs of the same experiment. To evaluate the variance in classification accuracy caused by different initial labeled data, we draw five random initial labeled sets ($L_0 \ldots L_4$) with replacement. Each of these five sets were used to train the base model, initialized with random weights, 5 times; a total of 25 models were trained

Experiments	>RSB	<RSB	Undetermined	Kruskal-Wallis P-Value
C10 (20%)	-	-	Coreset, DBAL, BALD, VAAL, QBC, UC	9.35×10^{-4}
C10 (30%)	-	VAAL	Coreset, DBAL, BALD, QBC, UC	2.67×10^{-9}
C10 (40%)	BALD, DBAL, QBC	UC	VAAL, Coreset	8.54×10^{-19}
C100 (20%)	Coreset, QBC	-	DBAL, BALD, UC	1.13×10^{-8}
C100 (30%)	QBC	VAAL	Coreset, DBAL, BALD, UC	1.84×10^{-12}
C100 (40%)	Coreset, DBAL, QBC	-	BALD, VAAL, UC	5.63×10^{-16}

Table 1. Statistical Analysis of variance; C10/100 refers to CIFAR 10/100

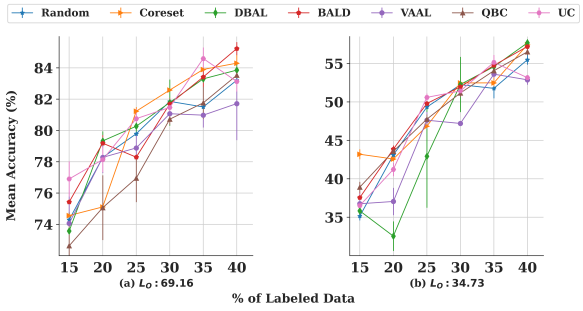

Figure 2. Results using 5% of training data is annotated at each iteration of AL on **(a)** CIFAR10 and **(b)** CIFAR100. Mean accuracy for the base model (at 10% labeled data) is noted at the bottom of each plot.

for each AL method to characterize variance within-sample-sets and between-sample-sets. From the results summarized in Fig. 1, we make the following observations: (i) A standard deviation of **1-2**% in accuracy among different AL methods, indicating that out of chance, it is possible to achieve seemingly better results, (ii) In contrast to previous studies, our extensive experiments indicate that no AL method performs consistently better, and random sampling baseline performs competitively. As stated previously, we believe that automatic hyper-parameter tuning repeated for each AL method at every AL iteration, while adding computation burden significantly, shows that there is marginal or no improvement achieved using AL methods compared to the random sampling baseline, (iii) Our results averaged over 25 runs in Fig. 1 (rightmost column) indicate that no AL method performs consistently best.

6.2. Statistical Analysis of Variance

In order to statistically compare the results achieved by the AL methods, we assessed the normality and variance to validate assumptions for parametric tests first. The normality assumption was tested using Kolmogorov-Smirnov test, and the assumption of homoscedasticity was tested using Levene's test. We found that normality could not be assumed for any of the six experiments in Tab. 1. Using Levene's test, we found that the null hypothesis of equality of variance was rejected at $\alpha = 0.05$ in 4 out of 6 experiments in Tab. 1. Therefore, for consistency, we used Kruskal–Wallis one-way ANOVA for all 6 experiments, and found that at least one method stochastically dominated one other method in all 6 experiments. Next, we used Games-Howell test for post-hoc multiple pairwise comparison to assess which pairs of methods differed significantly in their results. At a threshold of $\alpha = 0.05$, we observed that no AL method consistently outperforms random baseline. The null hypothesis of equal mean accuracy could not be rejected for most pairwise comparisons. We did note that among the methods evaluated, QBC was superior to RSB in 4 out of 6 experiments, with other methods performing as follows: Coreset 2/6, DBAL 2/6, BALD 1/6. Our analysis strongly suggests that variance in performance metric needs to be assessed to fairly compare an AL method against RSB.

6.3. Differing Experimental Conditions

Next, we compare AL methods and RSB by modifying different experimental conditions for annotation batch size, size of validation set, and class imbalance.

Annotation Batch Size (b): Following previous studies, we experiment with annotation batch size (b) equal to 5%, and 10% of the overall sample count ($L + U$). Results in Fig. 2 (corresponding Tab 6 in suppl.) show that our observation still holds i.e no AL method is consistently better. For example, on CIFAR10 at 40% labeled data and $b = 10\%$ (refer Fig. 1 and Tab 7. in suppl.), UC outperforms all other methods but when compared to 40% labeled data and $b = 5\%$ (refer Fig. 2 and Tab 6. in suppl.), BALD performs the best. Similarly on CIFAR100 at 30% labeled data and $b = 10\%$, RSB outperforms coreset but when compared to 30% labeled data and $b = 10\%$, coreset performs better. We therefore conclude that no AL method offers consistent advantage over others under different budget size settings.

Validation Set Size: During training, we select the best performing model on the validation set (V) to report the test set (T_s) results. To evaluate if size of V can affect the conclusions drawn from comparative AL experiments, we perform experiments on CIFAR100 with three different V sizes: 2%, 5%, and 10% of the total samples ($L + U$). From results in Tab. 2, we did not observe discernible trends in accuracy with respect to the size of V. For instance, RSB achieves a mean accuracy of 47.5%, 43.4%, and 46.7%, respectively, for the best model selected using 2%, 5% and 10% of the training data as V. However, when the labeled data increases the range of accuracy values is not as large, indicating that training tends to suffer from less variance when higher volume of labeled data was available. For example, at 40% labeled data the RSB achieves a mean accuracy of 54.58%, 57.24%, and 55.06%. From these experiments, it appears that the initial AL iterations were more sensitive to size of V compared to the later iterations. Furthermore, in line with previous experiment, no AL method was consis-

	2%			5%			10%		
Methods	20%	30%	40%	20%	30%	40%	20%	30%	40%
RSB	47.52 ± 0.58	52.83 ± 0.86	54.58 ± 0.56	43.43 ± 0.61	52.05 ± 0.6	57.24 ± 0.28	46.67 ± 0.3	51.43 ± 0.81	55.06 ± 0.35
Coreset	43.69 ± 0.53	**53.37 ± 0.87**	**57.41 ± 0.91**	44.31 ± 0.26	**54.01 ± 0.87**	56.98 ± 0.86	**47.33 ± 0.64**	49.73 ± 0.92	57.05 ± 0.4
DBAL	40.32 ± 0.45	50.63 ± 0.55	57.12 ± 0.45	43.48 ± 1.00	51.45 ± 0.6	57.36 ± 0.73	45.53 ± 2.33	51.04 ± 0.49	**58.06 ± 0.51**
BALD	46.61 ± 0.37	51.14 ± 0.60	56.11 ± 0.35	44.47 ± 0.74	51.35 ± 0.61	**58.18 ± 0.14**	47.1 ± 1.24	50.4 ± 0.88	55.65 ± 0.34
VAAL	42.57 ± 0.89	49.94 ± 1.24	54.28 ± 0.77	**46.12 ± 0.96**	51.27 ± 0.81	54.41 ± 0.59	39.73 ± 0.43	50.95 ± 0.88	55.23 ± 0.63
QBC	44.86 ± 0.57	51.81 ± 0.53	56.9 ± 0.639	44.64 ± 0.77	52.16 ± 0.38	57.02 ± 0.21	46.04 ± 0.57	**53.2 ± 0.38**	57.63 ± 0.49
UC	44.65 ± 1.14	51.79 ± 0.29	55.81 ± 0.49	43.76 ± 0.19	52.52 ± 0.93	57.66 ± 0.24	41.37 ± 1.29	52.97 ± 0.83	55.45 ± 0.62

Table 2. Test set performance for model selected with different validation set sizes on CIFAR100. Results are average of 5 runs.

tently better across AL iterations as size of V changes.

Class Imbalance: Here, we evaluate the robustness of different AL methods on imbalanced data. For this, we construct L_0 on CIFAR100 dataset, to simulate long tailed distribution of classes by following a power law, where the number of samples of 100 classes are given by $\text{samples}[i] = a + b * \exp(\alpha x)$ where $i \in \{1 \dots 100\}; a = 100, x = i + 0.5, \alpha = -0.046$ and $b = 400$. The resulting image count per class is normalized to construct a sampling distribution. Models were trained using previously described settings, with the exception of loss function which was set to weighted cross entropy.

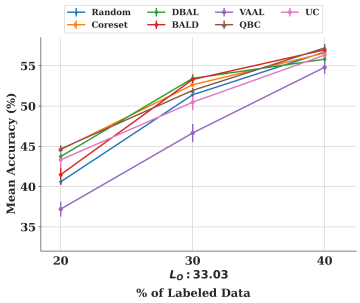

Figure 3. Results are average of 5 runs on imbalanced CIFAR100. The mean accuracy for the base model (at 10% labeled data) is noted at the bottom of plot.

The results in Fig. 3 show that the difference between RSB and the best AL method reduces with more and more labeled data. More importantly, we notice that AL methods demonstrate different degree of change in the imbalanced class setting, without revealing a clear trend in the plot. From clear trend, we mean that a robust AL method is expected to perform consistently best across all fractions of the data, which is not the case in Fig. 3.

6.4. Regularization

With the motivation stated in Sec. 3, we evaluate the effectiveness of advanced regularization techniques (RA and SWA) in the context of AL using CIFAR10 and CIFAR100 datasets. We refer the models trained using such advanced regularization techniques as *strongly-regularized* models (SR models) in further experiments. We empirically observed that unlike ℓ_2-regularization, which requires careful tuning, results using RA & SWA were fairly robust to changes in their hyper-parameters.

Fig. 5 compares different AL methods with RSB on CIFAR10/100 datasets. We observe that strongly-regularized models consistently achieve significant performance gains across all AL iterations and exhibit appreciably-smaller variance across multiple runs of the experiments. Our strongly-regularized random-sampling baselines on 40% labeled data achieves mean accuracy of 90.87% and 59.36% respectively on CIFAR10 and CIFAR100. We note that for CIFAR10, the RSB-SR model with 20% of training data achieves 3% higher accuracy compared to RSB model trained using 40% of the training data. Similarly for CIFAR100, the RSB-SR 30%-model performs comparably to the 40%-RSB model. Therefore, we consider techniques under strong regularization to be a valuable addition to the low-data training regime of AL, especially given that it significantly reduces the variance in evaluation metric which can help avoid misleading conclusions.

Methods	CIFAR10	CIFAR100
RSB	69.16	34.73
+ SWA	74.6	38.06
+ RA	76.36	38.01
+ Shake-Shake(SS)	73.93	39.23
+ SWA + RA	82.16	39.44
+ SS + SWA + RA	**84.45**	**48.92**

Table 3. Individual contributions of different regularization techniques. Results correspond to the best trial (total trials=50) found by AutoML using 10% of training data. Above experiments use the VGG16 except for Shake-Shake as it is restricted to the family of resnext.

An ablative study to show individual contribution of each regularization technique towards overall performance gain is given in Tab. 3. Results indicate that both RA & SWA show a significant combined gain of $\approx 12\%$ and $\approx 5\%$ on CIFAR10 and CIFAR100 respectively. We also experimented with Shake-Shake (SS) [8] in parallel to RA & SWA, and observed that it significantly increases the runtime, and is not robust to model architectures. We therefore chose RA & SWA over SS for strongly-regularized models.

6.5. Active Learning on ImageNet

Compared to CIFAR10/100, ImageNet is more challenging with larger sample count, 1000 classes and higher resolution images. In order to work with available compute resources, we compared coreset, VAAL and RSB on ImageNet. The details for training hyper-params are in supplementary material. Results with and without strong regular-

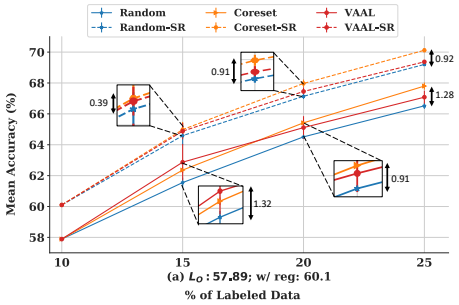

Figure 4. Effect of strong regularization (RA, SWA) (shown in dashed lines) on Imagenet where annotation budget is 5% of training data. Reported results are averaged over 3 runs. For exact accuracies we refer readers to the Tab 5. in suppl.

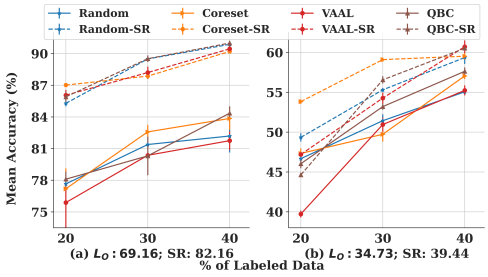

Figure 5. Effect of strong regularization (RA, SWA) on the test accuracy of CIFAR10(a) and CIFAR100(b). The mean accuracy for the base model (at 10% labeled data) is noted at the bottom of each plot.

	Source Architecture			Target Architectures								
	VGG16			WRN-28-2			R18			R18-SR		
Methods	20%	30%	40%	20%	30%	40%	20%	30%	40%	20%	30%	40%
RSB	77.34	80.91	82.05	81.3	81.98	85.68	75.73	79.69	80.69	88.69	90.44	91.8
Coreset	76.18	82.24	85.11	**82.32**	84.29	86.72	**79.27**	**82.13**	**85.09**	88.48	91.72	92.94
DBAL	78.08	82.42	85.17	80.79	84.54	87.87	75.05	80.69	82.95	**89.41**	**92.23**	**93.34**
VAAL	77.13	79.97	80.55	78.42	83.79	86.58	74.87	79.96	82.24	87.11	90.43	92.19
QBC	**78.63**	**82.66**	**85.24**	81.85	**85.06**	**87.94**	76.61	81.77	84.41	88.34	91.37	92.62

Table 4. Transferability experiment on CIFAR10 dataset where source model is VGG16 and target model is Resnet18 (R18) and Wide Resnet-28-2 (WRN-28-2). Test accuracies are reported corresponding to the best model trained on CIFAR10/100 dataset. For best model hyper-parameters we perform random search over 50 trials. Results with strong regularization is shown in the last column.

Methods ↓	10%	20%	30%	40%
Noise: 10%				
RSB	69.16	72.08	76.62	80.88
RSB-SR	82.16	84.96	86.06	89.13
Noise: 20%				
RSB	69.16	69.42	75.89	79.61
RSB-SR	82.16	77.39	85.9	85.12

Table 5. RSB accuracy with and without strong regularization on CIFAR10 with noisy oracle.

ization are shown in Fig. 4 (refer Tab 4 in suppl.) where mean accuracies are reported over 3 runs. Using ResNext-50 architecture [29] and following the settings of [34], we achieve improved baseline performances compared to the previously reported results [2, 27]. From Fig. 4, we observe that VAAL outperforms all other methods at 15% data but both Coreset and Coreset-SR leads to the highest mean accuracy across other settings. We also note that similar to the observations on CIFAR datasets; (i) strong regularization helps improve performance across the methods and fraction of data, and (ii) it also reduces the performance gap between RSB and other AL methods. For example, RSB-SR model with 20% data exceeds the performance achieved by both RSB and VAAL using 25% data, for a saving of $\approx$ 64000 images with labels.

6.6. Transferability Settings

In this experiment, we evaluated how does the accuracy compare when an active learning method chosen for one task architecture (e.g. VGG16), is used to train another task model (e.g. ResNet18). We conduct an experiment by storing the indices of sample set drawn in an AL iteration on the source network (VGG16), and use them to train the target network (ResNet18 and WRN-28-2). From Tab. 4, we observe the inconsistency in performance of AL methods. For example, on CIFAR10 with Resnet18, Coreset achieves consistently best performance. However, this observation does not hold true for strongly-regularized models. We also note that for ResNet18 target architecture, the RSB-SR model outperforms the best AL approach (coreset with ResNet18) in all the AL iterations, though DBAL-SR appears at the top with an accuracy of 93.34% at 40% labeled data. We therefore conclude that the AL methods are sensitive to the model architecture being used.

7. Additional Experiments

Noisy Oracle: In this experiment, we sought to evaluate the stability of strongly regularized network to labels from a noisy oracle. We experimented with two levels of oracle noise by randomly permuting labels of 10% and 20% of samples in the set drawn by random sampling baseline at each iteration. From results in Tab. 5, we found that the drop in accuracy for the strongly-regularized model regularized was nearly half (3%) compared to its complement model trained (6%) on both 30% and 40% data splits. Our findings suggest that the noisy pseudolabels generated for the unlabelled set U by model ϕ, when applied in conjunction with appropriate regularization, should help improve model's performance. Additional results using AL methods in this setting are shared in the supplementary material.

Active Learning Sample Set Overlap: For interested readers, we discuss the extent of overlap among the sample sets drawn by AL methods in supplementary.

Optimizer Choices: Different AL studies have reported

different optimizer choices in their experiments. In this light, we analyze the optimizer chosen by AutoML and we analyze it on CIFAR10. The results are present in supplementary section. Contrary to the previous studies where the optimizer is fixed in advance, we found that both adam and sgd optimizer can sometimes work better than the other.

8. Discussion

Under-Reported Baselines: We note that several recent AL studies show baseline results that are lower than the ones reproduced in this study. Tab 1 in the supplementary summarizes our RSB results with comparisons to RSB reported by some of the recently published AL methods, under similar training settings. Based on this observation, we emphasize that comparison of AL methods must be done under a consistent set of experimental settings. Our observations confirm and provide a stronger evidence for a similar conclusion drawn in [22], and to a less related extent, [23]. Different from [22] though, we demonstrate that: **(i)** Relative gains using AL method are found under a narrow combination of experimental conditions. **(ii)** Fixing the hyper-parameters at the start of AL can result into sub-optimal results. **(iii)** More distinctly, we show that performance gains (when they exist) are significantly lower for *strongly-regularized* models.

The Role of Regularization: Regularization helps reduce generalization error and is particularly useful in training overparameterized neural networks with low data. We show that both RA & SWA can achieve appreciable gain in performance at the expense of a small computational overhead. We observed that along with learning rate (in case of SGD), regularization was one of the key factors in reducing the error while being fairly robust to its hyperparameters (in case of RA and SWA). We also found that the margin of the gain observed with an AL method over RSB on CIFAR10/100 significantly minimize when the model is well-regularized. Strongly-regularized models also exhibited smaller variance in evaluation metric. With these observations, we recommend that AL methods be also tested using well-regularized model to ensure their robustness. Lastly, we note that there are multiple ways to regularize the data-model-metric pipeline, we focus on data and model side regularization using techniques such as RA and SWA, though it is likely that other combination of newer regularization techniques will lead to similar results. We do believe that with their simplicity and applicability to a wide variety of model (compared to shake-shake method), RA & SWA can be effectively used in AL studies to reduce the variance.

AL Methods Compared To Strong RSB: In contrast to previous findings, well-regularized random baseline in our study was either at par with or marginally inferior to the state-of-the-art AL methods. We believe that previous studies that ran a comparison against the random baseline might have insufficiently regularized the models and/or did not tune the hyperparameters. We also observed (Tab. 4) that a change in model architecture can change the conclusions being drawn in comparing an AL method to a random baseline. This observations suggests that either transferability experiments should be conducted to assess if the active sets are indeed useful across architectures, or comparisons are repeated with additional architectures to study the robustness of an AL method to changes in architecture. Similarly we observed that the strong regularization achieves two goals: (i) it helps improve the performance, (ii) more importantly, it helps reduce the variance in results and reduces the performance gap between random baseline and other AL methods. The highly-sensitive nature of AL results using neural networks therefore necessitates a comprehensive suite of experimental tests.

9. Conclusion and Proposed Guidelines

Our extensive experiments suggest a strong need for a common evaluation platform that facilitates robust and reproducible development of AL methods. To this end, we recommend the following to ensure results are robust:

(i) Experiments should be repeated under varying training settings such as model architecture and budget size, among others. **(ii)** Regularization techniques such as RA & SWA should be incorporated into the training to ensure AL methods are able to demonstrate gains over a strong-regularized random baseline.**(iii)** Transferability experiments should be performed to test how useful AL-drawn sample sets are. Alternatively, experiments should be repeated using multiple architectures.**(iv)** To increase the reproducibility of AL results, experiments should ideally be performed using a common evaluation platform under consistent settings to minimize the sources of variation in the evaluation metric. **(v)** Snapshot of experimental settings should be shared, e.g. using a configuration file (.cfg, .json etc). **(vi)** Index sets for a public dataset used for partitioning the data into training, validation, test, and AL-drawn sets should be shared, along with the training scripts.

In order to facilitate the use of these guidelines in AL experiments, we provide a python-based AL toolkit. We provide the index sets for the datasets used in this study that was used to partition the data into training, validation, and test sets. Lastly, all experiment configuration files are also shared as part of the toolkit.

Societal impact and Limitations: For some of our experiments, we use ImageNet data, which lacks gender and ethnic diversity [30]. The models learned with this data could therefore have biased representations. Further, ImageNet has privacy concerns due to unblurred faces [31]. A limitation of our work is that all our experiments are conducted for image classification. We leave other tasks such as detection and segmentation for future work.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019. 3

[2] William H. Beluch, Tim Genewein, Andreas Nürnberger, and Jan M. Köhler. The power of ensembles for active learning in image classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018. 1, 2, 3, 7

[3] Razvan Caramalau, Binod Bhattarai, and Tae-Kyun Kim. Sequential graph convolutional network for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9583–9592, 2021. 1

[4] 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. 3

[5] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019. 3

[6] Melanie Ducoffe and Frédéric Precioso. Adversarial active learning for deep networks: a margin based approach. *CoRR*, abs/1802.09841, 2018. 1, 2

[7] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org, 2017. 1, 2

[8] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 3, 6

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 2

[10] 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. 4

[11] Siyu Huang, Tianyang Wang, Haoyi Xiong, Jun Huan, and Dejing Dou. Semi-supervised active learning with temporal output discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3447–3456, 2021. 2

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4

[13] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 3

[14] Kwanyoung Kim, Dongwon Park, Kwang In Kim, and Se Young Chun. Task-aware variational adversarial active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8166–8175, 2021. 1, 2

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[16] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *CoRR*, abs/1906.08158, 2019. 1, 2

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 1

[18] Kuangliu. Resnet18.pytorch. https://github.com/kuangliu/pytorch-cifar, 2017. 4

[19] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994. 2

[20] David Lowell, Zachary C. Lipton, and Byron C. Wallace. How transferable are the datasets collected by active learners? *CoRR*, abs/1807.04801, 2018. 1

[21] Meliketoy. wide-resnet.pytorch. https://github.com/meliketoy/wide-resnet.pytorch, 2017. 4

[22] Sudhanshu Mittal, Maxim Tatarchenko, Özgün Çiçek, and Thomas Brox. Parting with illusions about deep active learning, 2019. 1, 8

[23] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018. 8

[24] Ameya Prabhu, Charles Dognin, and Maneesh Singh. Sampling bias in deep active classification: An empirical study. *arXiv preprint arXiv:1909.09389*, 2019. 1

[25] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. 1, 2, 3

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 4

[27] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. *arXiv preprint arXiv:1904.00370*, 2019. 1, 2, 3, 7

[28] Toan Tran, Thanh-Toan Do, Ian D. Reid, and Gustavo Carneiro. Bayesian generative active deep learning. *CoRR*, abs/1904.11643, 2019. 1

[29] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 4, 7

[30] Kaiyu Yang, Klint Qinami, Li Fei-Fei, Jia Deng, and Olga Russakovsky. Towards fairer datasets: Filtering and balancing the distribution of the people subtree in the imagenet hierarchy. pages 547–558, 2020. 8

[31] Kaiyu Yang, Jacqueline Yau, Li Fei-Fei, Jia Deng, and Olga Russakovsky. A study of face obfuscation in imagenet. *arXiv preprint arXiv:2103.06191*, 2021. 8

[32] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 93–102, 2019. 1, 2

[33] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 4

[34] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S^4l: Self-supervised semi-supervised learning. *CoRR*, abs/1905.03670, 2019. 3, 4, 7

[35] Beichen Zhang, Liang Li, Shijie Yang, Shuhui Wang, Zheng-Jun Zha, and Qingming Huang. State-relabeling adversarial active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8756–8765, 2020. 2