Towards Comparable Active Learning

David S. Hippocampus*
Department of Computer Science
Cranberry-Lemon University
Pittsburgh, PA 15213
hippo@cs.cranberry-lemon.edu

Abstract

The abstract paragraph should be indented $\frac{1}{2}$ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

Version: 1.0

Active Learning (AL) plays an important role in our society that applies machine learning to more and more areas and therefore has a high demand for labeled data in more and more areas. A problem that concerns academic researchers and practitioners in businesses alike and even could be extended to education in schools and hobbyists around the world. On top of providing a principled way to labeled unlabeled datasets, active learning is one of the two major approaches besides semi-supervised learning to make deep learning models more data efficient by requiring only a limited set of manually labeled data. Both approaches are at their core orthogonal and can freely be combined and therefore we should continue our research efforts for both approaches.

Among others, the authors of [14] have pointed out severe inconsistencies in results of AL papers in recent years. In their supplementary materials they conducted a meta analysis of reported results of several different AL algorithms and found that all considered algorithms only provided significant lifts in their own original papers, while all following literature reported performances no better that uncertainty sampling, or in some cases no better than random sampling for the same algorithm. The result of these inconsistencies is a chaotic landscape of AL algorithms where every paper claims to archive state-of-the-art results by significantly outperforming everyone else, while the vast majority of results proves to be non-reproducible.

1.1 Contributions

1. Evaluation of Active Learning algorithms on datasets from 4 different domains, including synthetic data that highlights principled shortcomings of existing approaches.
2. Novel experimental protocol for seeding the experiment with 3 different seeds to allow full control and reproducibility and analysis of how many restarts are required to converge to the true median performance reliably.
3. Simple algorithm for an Oracle-Curve that can be constructed greedily and does not rely on search.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Submitted to the 37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks. Do not distribute.

2 Related Work

Version: Braindump

Many different algorithms have been proposed for active learning. In this work we focus on those approaches that have shown consistent results over the years as well as newer approaches that have demonstrated significant lifts in their initial experiments. AL algorithms can be categorized into two classes: Geometric approaches and uncertainty-based approaches. Geometric approaches include CoreSet [12] and TypiClust [4], which use clustering techniques to partition the data and then sample their unlabeled points based on the clusters. Uncertainty-based approaches include classic uncertainty sampling (based on Shannon-Entropy and the margin-score), BALD [6] and BADGE [1], which use metrics to measure the classifiers state.

Some previous work also aimed to provide a benchmark suite for active learning: The authors of [2] and [8] both focus on active learning in the image domain. While [2] discuss a new metric to measure AL performance, which they call "Label Efficiency" and provide experiments on many common configurations for data preparation, model training and other hyperparameters, [8] focuses on combined approaches of AL and semi-supervised learning to aid model training. The authors of [5] study models that are learned with AL techniques in the image and text domain. They test for several different properties of the models including robustness, response to compression techniques and final performance.

3 Overview

Table 1 shows a feature comparison between our proposed benchmark and several existing benchmarks in the literature, as well as methodological AL papers with an extensive experiments section. TODO We include in this table any methodological paper that experiments on at least two domains. TODO Define AL scenarios (really hard)

Paper	# Datasets	Domains	Scenarios	Oracle	RL Alg.
Beck et al. [2]	4	1	3	-	-
Hu et al. [5]	5	2	1	-	-
Li et al. [8]	5	1	1	-	-
Zhou et al. [14]	3	2	1	✓	-
Ours	5	2	2	✓	-

Table 1: Comparison of our benchmark with the existing literature

3.1 Problem Description

Version: 1.0

Given two spaces $\mathcal{X} := \mathcal{R}^M$ and $\mathcal{Y} := \mathcal{R}^C$, a sample $\mathcal{D}_1, \dots, \mathcal{D}_N \subseteq (\mathcal{X} \times \mathcal{Y})^*$ of sequences of pairs (x, y) from an unknown distribution p called datasets and a number $B \in \mathcal{N}$ with $B < |\mathcal{D}|$. Given two functions $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathcal{R}$ called loss, and $A : (\mathcal{X} \times \mathcal{Y})^* \times \mathcal{X}^* \to \mathcal{Y}^{\mathcal{X}}$ called learning algorithm, find a function

$$a : (\mathcal{X} \times \mathcal{Y})^* \times \mathcal{X}^* \to \{0, 1\}^* \qquad \text{(equivariant in the second argument)}$$

called acquisition function, s.t. the expected loss of a model learned on all predictors plus B acquired targets is minimal:

$$\begin{aligned} &\mathbb{E}_{\mathcal{D}\text{train},\mathcal{D}\text{test}\sim p}\, \text{avg}_{(x,y)\in\mathcal{D}\text{test}}\, \ell(y, \hat{y}(x)) \\ &\quad \text{with } \hat{y} := A((\mathcal{D}\text{train}_{n_1}, \dots, \mathcal{D}\text{train}_{n_B}), \mathcal{D}\text{train}|_{\mathcal{X}}) \\ &\qquad n_b := \text{index}(a((\mathcal{D}\text{train}_{n_1}, \dots, \mathcal{D}\text{train}_{n_{b-1}}), \mathcal{D}\text{train}|_{\mathcal{X}})), \quad b \in 1{:}B \end{aligned}$$

(*We would need to switch from lowest expected loss to highest AUC*)

Since combinatorial problem of finding the optimal subset $\mathcal{D}_{\text{train}}$ is computational not feasible, we

allow sequential construction of $\mathcal{D}_{\text{train}}$.

To construct the active learning setting from existing labeled datasets, we first split it into $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$ and then suppress the labels $\mathcal{Y}$ of $\mathcal{D}_{\text{train}}$ to form the unlabeled pool $\mathcal{U} := \mathcal{X}^*$ and form and initial labeled pool by uniformly sampling k number of instances per class from $\mathcal{U}$ and recovering their label $\mathcal{L} := (\mathcal{X}, \mathcal{Y})^{k*C}$.

Following [14], the quality of an active learning algorithm is evaluated by an "anytime" protocol that incorporates classification performance at every iteration, not just the final performance after the budget is exhausted. We employ the normalized area under the accuracy curve (AUC):

$$\text{auc}(\mathcal{U}, \mathcal{L}, \hat{y}, B) := \frac{1}{B} \sum_{i=1}^{B} \text{Acc}(y_{test}, \hat{y}_i(x_{test})) \tag{1}$$

Where $\hat{y}_i$ is the retrained classification model after the i-*th* instance was selected.

Algorithm 1 Oracle

Require: $\mathcal{U}$ ▷ Unlabeled Pool
Require: τ ▷ Unlabeled Sample Size
$\mathcal{L}^{(1)} \leftarrow \text{seed}(\mathcal{U})$
$\mathcal{U}^{(1)} \leftarrow \mathcal{U} \setminus \mathcal{L}^{(1)}$
for $i := 1 \dots B$ **do**
  $\text{acc}^{(i)} \leftarrow \text{Retrain}(\mathcal{L}^{(i)})$
  $u^{(i)} \overset{\tau}{\sim} \text{unif}(1 : |\mathcal{U}^{(i)}|)$
  $r* \leftarrow 0$
  for $j := 1 \dots \tau$ **do**
    $y^{(j)} \leftarrow \text{label}(u_j^{(i)})$
    $\mathcal{L}^{(j)} \leftarrow \mathcal{L}^{(i)} \cup \{(u_j^{(i)}, y^{(j)})\}$
    $\text{acc}^{(j)} \leftarrow \text{Retrain}(\mathcal{L}^{(j)})$
    $r^{(j)} \leftarrow \text{acc}^{(j)} - \text{acc}^{(i)}$
    if $r^{(j)} > r^*$ **then**
      $r* \leftarrow r^{(j)}$
      $j* \leftarrow j$
    end if
  end for
  if $r^* = 0$ **then**
    $j* \leftarrow \text{margin}(u^{(i)}, \hat{y}_\theta)$
  end if
  $y^{(i)} \leftarrow \text{label}(u_{j*}^{(i)})$
  $\mathcal{L}^{(i+1)} \leftarrow \mathcal{L}^{(i)} \cup \{(u_{j*}^{(i)}, y^{(i)})\}$
  $\mathcal{U}^{(i+1)} \leftarrow \mathcal{U}^{(i)} \setminus \{u_{j*}^{(i)}\}$
end for
return $\frac{1}{B} \sum_{i=1}^{B} \text{acc}^{(i)}$

Algorithm 2 Retrain

Require: $\mathcal{L}$ ▷ Labeled Pool
Require: $\mathcal{D}_{\text{test}}$ ▷ Test Data
Require: $\hat{y}_\theta$ ▷ Class. Model
Require: e^{max} ▷ Maximum Epochs
$\text{loss}^* \leftarrow \infty$
for $i := 1 \dots e^{\text{max}}$ **do**
  $\theta_{i+1} \leftarrow \theta_i - \eta \nabla_\theta \ell(\mathcal{L}, \hat{y}_\theta)$
  $\text{loss}_i \leftarrow \ell(\mathcal{L}^{\text{test}}, \hat{y}_\theta)$
  if $\text{loss}_i < \text{loss}^*$ **then**
    $\text{loss}^* \leftarrow \text{loss}_i$
  else
    Break
  end if
end for
return $\text{Acc}(\mathcal{L}^{\text{test}}, \hat{y}_\theta)$

In Alg. 1 $\text{Retrain}(\mathcal{L}^{(i)})$ is shorthand for $\text{Retrain}(\mathcal{L}^{(i)}, \mathcal{D}_{\text{test}}, \hat{y}_\theta, e^{\text{max}})$, $\text{label}(u_j^{(i)})$ recovers the label of instance $u_j^{(i)}$ and $\text{margin}(u^{(i)}, \hat{y}_\theta)$ applies margin sampling to select as a fallback, when the oracle has not found another option that induces a positive change in classification performance.

4 Methodology

4.1 Evaluation

Version: 1.0

We compare different AL algorithms based on their median AUC score (Eq. 1) across multiple restarts of the experiment. Each restart will retain the train/test split (often given by the dataset itself), but introduces a new validation split to mimic the leave-one-out protocol for cross-validation. The AUC incorporates performance in early stages (low budget) as well as capabilities to push the classifier in later stages (high budget). A good AL algorithm should be able to perform well in both scenarios.

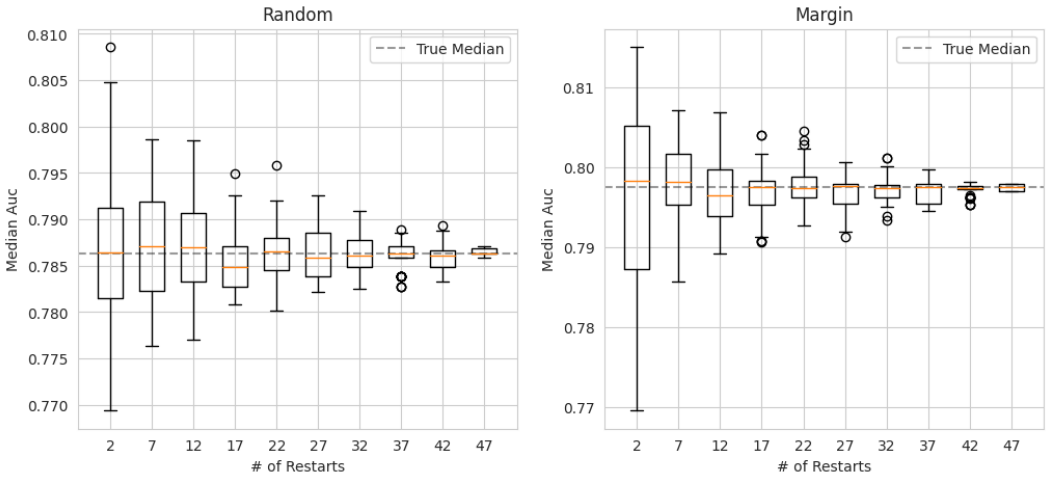


Figure 1: Random draws from Splice with different numbers of repetitions.

Since AUC is dependent on the chosen budget, we need a general rule on how to set this hyperparameter that does not inherently benefit a subset of algorithms. In this work, we choose the budget per dataset to be the first point at which any algorithm (except oracle) manages to reach a percentage of the upper bound performance measured on the full dataset. Even though we would like to propose a single percentage value for all datasets, we found that different data modalities and use cases need different percentages to produce sensible budgets. We propose the following values: **Tabular**: 99%, **Image**: 90% and **Text**: 95%.

Additionally, we provide evidence in Fig. 1 that previous works have not evaluated their experiments with a sufficient number of restarts. To create Fig. 1 we used all our 50 runs from the margin/random sampling algorithm on the splice dataset. From these 50 runs we uniformly sampled subsets of runs and calculated the median AUC for this subset. One AUC value like this corresponds to one cross-validated experiment sampled from the distribution of experiments that are restarted exactly this many times. To create one slice in Fig. 1, we drew 50 samples from this distribution. Each box-plot represents the variance of an evaluation if conducted with the respective number of restarts. We can clearly observe that low repetitions (< 10) provide an uncertain evaluation where lucky and unlucky draws of the same experiment give drastically different median AUC values. To combat is uncertain evaluation, we propose to repeat every experiment 50 times, to arrive at the true median AUC for each algorithm.

4.2 Reproducibility

Version: 1.0

A big focus in this work is to provide an experimental setup that is fully reproducible independent of the dataset, classification model or AL algorithm used. In our opinion, an evaluation on one dataset with a given seed should always be done on the same validation split as well as using the same random state for all included systems like the mini batch sampler for the classifier model training. Even though different AL algorithms will pick different samples, making them unavailable for sampling in earlier or later batches, the theoretical decision tree for every possible choice in every iteration i should stay the same. Since every possible trajectory cannot be precomputed and stored to disk, we need to resort to seeding the evaluation. The default choice of setting a global seed at the start of the experiment is not sufficient here, since a single additional random draw from the random number generator completely changes the behavior of all other systems. This additional random number might be drawn during the initialization of the classification model, the AL algorithm, or even during every AL iteration if Ω is stochastic. This problem also applies to the initialization

of the classification model θ and the initialization and querying of Ω. The desired control over all these processes can only be archived by assigning a separate random number generator to all these processes. To the best of our knowledge, we are the first work that discusses this issue and proposes a solution for it. We hypothesize that the insufficient setup with global seeds contributes to the ongoing problem of inconsistent results of AL algorithms in different papers.

In summary, we introduce three different seeds: s_Ω for the acquisition function, $s_\mathcal{D}$ for dataset splitting and mini batch sampling and s_θ for model initialization and sampling of dropout masks. Unless stated otherwise, we will keep s_Ω fixed for restarts of the same experiment, while $s_\mathcal{D}$ and s_θ are incremented by 1 between restarts to introduce stochasticity into our framework.

4.3 Oracle

Version: 1.0

Posing active learning as a sequence ordering problem, the oracle sequence for a given combination of dataset, model and training procedure would be the sequence that induces the highest AUC score for a given budget. However, since this combinatorial problem is not solvable for realistic datasets, previous works have proposed approximations to this oracle sequence. [14] has used simulated annealing to search for the optimal sequence and used the best solution found after a fixed time budget. Even though their reported performance curves display a significant lift over all other algorithms, we found the computational cost of reproducing this oracle for all our datasets to be prohibitive (The authors reported the search to take several days per dataset on 8 V100 GPUs).

In this paper we propose a greedy oracle algorithm that constructs an approximation of the optimal sequence in an iterative fashion. Our oracle simply tests every data point in the provided sample of unlabeled points by fitting the classifier and directly measuring the resulting validation performance. The point with the best validation performance is selected and added to the labeled pool for that iteration. We noticed that this oracle is overfitting on the validation set, resulting in stagnating or even decreasing performance curves in later AL iterations. To circumvent this problem, we introduced margin sampling as a fallback option for the oracle. Whenever the oracle does not find an unlabeled point that results in an increase in performance (indicating an overfitting position), it defaults to margin sampling in that iteration. The pseudocode for our oracle can be found in Alg. 1.

5 Implementation Details

5.1 Sampling Strategies

Uncertainty Sampling

BALD

BADGE

Coreset

TypiClust

5.1.1 Honorable Mentions

Learning Loss for AL Introduces an updated classification model training with an auxiliary loss and therefore cannot be compared fairly against classification models without this boosted training regime.

5.2 Choosing the Classifier

Traditionally, the classifier is chosen per dataset so that it is capable of solving the dataset close to the SOTA performance reported in the literature. Similar to our hypothesis in section 4.1 we

Dataset	Architecture	Optimizer	LR	Weight Decay	Dropout
Splice	[24, 12]	NAdam	0.0012	5.9e-5	0
DNA	[24, 12]	NAdam	0.0398	3.6e-5	0
USPS	[24, 12]	Adam	0.0081	1.5e-6	0
FashionMnist	linear	-	-	-	-
Cifar10	linear	-	-	-	-

Table 2: Classifier architectures and optimized hyperparameters per dataset. Numbers in brackets signify a MLP with corresponding hidden layers.

hypothesize that AL algorithms will perform similarly on small classifiers and more complex ones, so that the overall ranking of algorithms stays the same. TODO: Transform this into full hypothesis incl. formatting?
On the basis of this hypothesis we opt to use smaller classifiers that still solve the dataset to a reasonable degree. Smaller classifiers also require fewer labeled datapoints to acquire performance close to the upper bound. This behavior can be observed in Figure TODO

For every dataset the chosen architecture's hyperparameters are optimized by to archive maximum upper bound performance. For an overview of architectures and hyperparameters please refer to Table 2.

5.3 Training the Classifier

Version: 0.1 Full rework needed
Generally, the classification model can be trained in two ways. Either you reset the parameters after each AL iteration and train the classifier from scratch with the updated labeled set $\mathcal{L}^{(i)}$, or you retain the previous state and only fine-tune the classifier on $\mathcal{L}^{(i)}$ for a reduced number of epochs. In this work we use the fine-tuning method for raw datasets to save computation, while we use the from-scratch training for embedded dataset, since they have very small classifiers and this method generally produces better results. Our fine-tuning scheme always trains for at least one epoch and employs an aggressive early stopping after that. The early stopping has patience 0, so it will stop as soon as the validation loss does no longer decrease.

6 Experiments

6.1 Datasets

Version: 1.0
For all our datasets we use the pre-defined train / test splits, if given. In the remaining cases, we define test sets upfront and store them into separate files to keep them fixed across all experiments. The validation set is split during experiment-time and depends on the dataset-seed.

Tabular We use **Splice**, **DNA** and **USPS** from LibSVMTools [10]. All three datasets are normalized between [0, 1].

Image We use **FashionMNIST** [13] and **Cifar10** [7]. Both datasets are normalized between [-1, 1].

Text We use **News Category** [9] and **TopV2** [3]. For News Category we use the 15 most common categories as indicated by its Kaggle site. We additionally drop sentences above 80 words to reduce the necessary padding (retaining 99,86% of the data). For TopV2, we are only using the "alarm" domain. Both datasets are encoded with pre-trained GloVe embeddings [11]. Since neither set provided a fixed test set, we split random 5000 datapoints into a test set.

We would like to point out that these datasets can be considered "toy-datasets" and therefore not relevant for practical purposes. This might be true if we aimed to develop novel classification models

	Splice	DNA	USPS
Oracle	0.830 +- 0.01	0.836 +- 0.02	0.823 +- 0.01
SAL	0.799 +- 0.01	0.797 +- 0.03	0.809 +- 0.01
Coreset	0.800 +- 0.01	0.795 +- 0.03	0.787 +- 0.02
TypiClust	0.790 +- 0.01	0.771 +- 0.04	0.761 +- 0.02
MarginScore	0.797 +- 0.02	0.795 +- 0.04	0.808 +- 0.01
ShannonEntropy	0.799 +- 0.02	0.794 +- 0.04	0.807 +- 0.01
RandomAgent	0.788 +- 0.01	0.765 +- 0.03	0.772 +- 0.01
Badge	0.807 +- 0.01	0.769 +- 0.06	0.797 +- 0.02
BALD	0.811 +- 0.01	0.743 +- 0.04	0.717 +- 0.05

Table 3: AUC values for all algorithms on the tabular datasets. Higher is better.

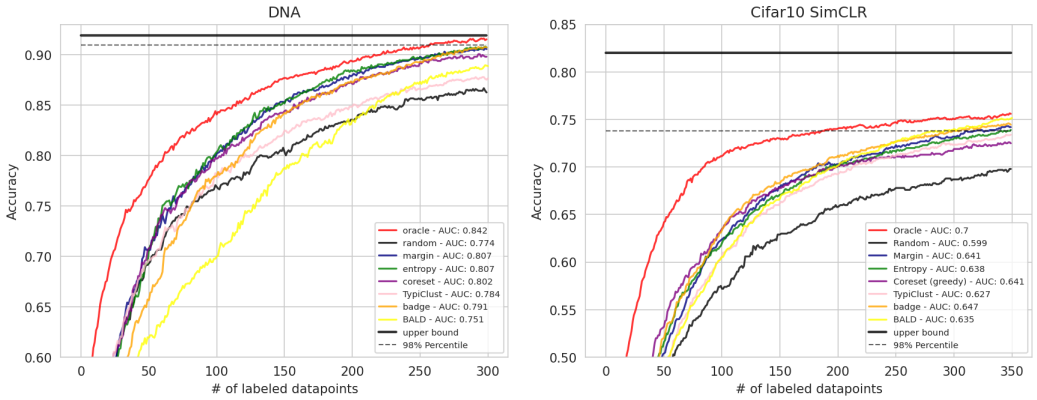


Figure 2: Results for all algorithms on two datasets

on these datasets, however we are solely focused on comparing different AL algorithms in this paper. Our core assumption is that a well-performing algorithm in our benchmark will transfer well into more practical use-cases.

Adapting the experimental setting from [4] we offer all our datasets in the raw setting as well as pre-encoded by a fixed embedding model that was trained by unsupervised contrastive learning. The text datasets are an exception, as they are only offered in their encoded form. The pre-encoded datasets offer multiple advantages in that they require smaller classification models (i.e. small MLPs) and in general much less sampled datapoints to archive close to upper bound performance (measured with the full dataset). They unify experimental setups for different datasets and therefore make them much more comparable.

6.2 Splice Embedded

7 Ablation Studies

- Setting τ to $|\mathcal{U}|$

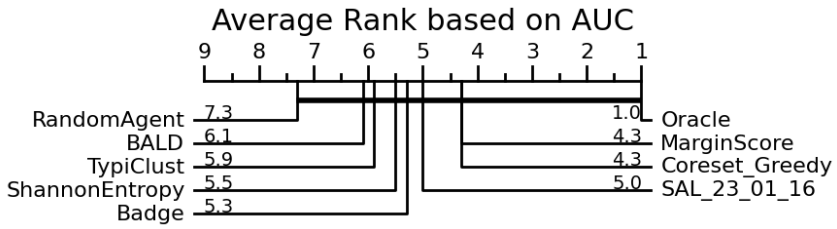


Figure 3: Critical Difference Diagram for all datasets and algorithms

- Reduction of the test set for speed

References

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

[2] Nathan Beck, Durga Sivasubramanian, Apurva Dani, Ganesh Ramakrishnan, and Rishabh Iyer. Effective evaluation of deep active learning on image classification tasks. *arXiv preprint arXiv:2106.15324*, 2021.

[3] Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020.

[4] Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. *arXiv preprint arXiv:2202.02794*, 2022.

[5] Qiang Hu, Yuejun Guo, Maxime Cordy, Xiaofei Xie, Wei Ma, Mike Papadakis, and Yves Le Traon. Towards exploring the limitations of active learning: An empirical study. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 917–929. IEEE, 2021.

[6] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32, 2019.

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Yu Li, Muxi Chen, Yannan Liu, Daojing He, and Qiang Xu. An empirical study on the efficacy of deep active learning for image classification. *arXiv preprint arXiv:2212.03088*, 2022.

[9] Rishabh Misra. News category dataset. *arXiv preprint arXiv:2209.11429*.

[10] Information Engineering Graduate Institute of Taiwan University. Libsvmtools.

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[12] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[13] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[14] Yilun Zhou, Adithya Renduchintala, Xian Li, Sida Wang, Yashar Mehdad, and Asish Ghoshal. Towards understanding the behaviors of optimal deep active learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1486–1494. PMLR, 2021.

A Other result tables

	SpliceEncoded	DNAEncoded	USPSEncoded
Oracle	0.754 +- 0.02	0.726 +- 0.02	0.674 +- 0.01
SAL	0.675 +- 0.03	0.640 +- 0.04	0.634 +- 0.01
Coreset	0.690 +- 0.02	0.644 +- 0.05	0.607 +- 0.02
TypiClust	0.695 +- 0.02	0.660 +- 0.03	0.643 +- 0.01
MarginScore	0.675 +- 0.03	0.643 +- 0.05	0.632 +- 0.02
ShannonEntropy	0.673 +- 0.03	0.638 +- 0.05	0.626 +- 0.02
RandomAgent	0.680 +- 0.03	0.633 +- 0.04	0.594 +- 0.02
Badge	0.670 +- 0.04	0.600 +- 0.07	0.597 +- 0.02
BALD	0.660 +- 0.04	0.597 +- 0.06	0.652 +- 0.01

Table 4: AUC values for all algorithms on the encoded tabular datasets. Higher is better.

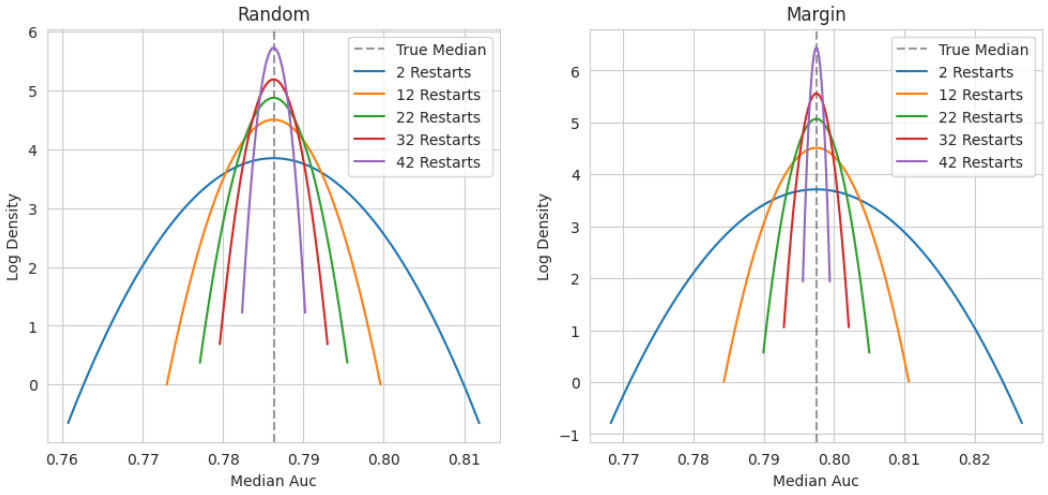
	Cifar10Encoded	FashionMnistEncoded
Oracle	0.699 +- 0.01	0.721 +- 0.01
SAL	0.623 +- 0.01	0.667 +- 0.01
Coreset	0.641 +- 0.01	0.674 +- 0.01
TypiClust	0.627 +- 0.01	0.638 +- 0.01
MarginScore	0.641 +- 0.01	0.676 +- 0.01
ShannonEntropy	0.637 +- 0.02	0.680 +- 0.01
RandomAgent	0.598 +- 0.02	0.633 +- 0.01
Badge	0.645 +- 0.02	0.681 +- 0.01
BALD	0.633 +- 0.01	0.666 +- 0.01

Table 5: AUC values for all algorithms on the encoded image datasets. Higher is better.

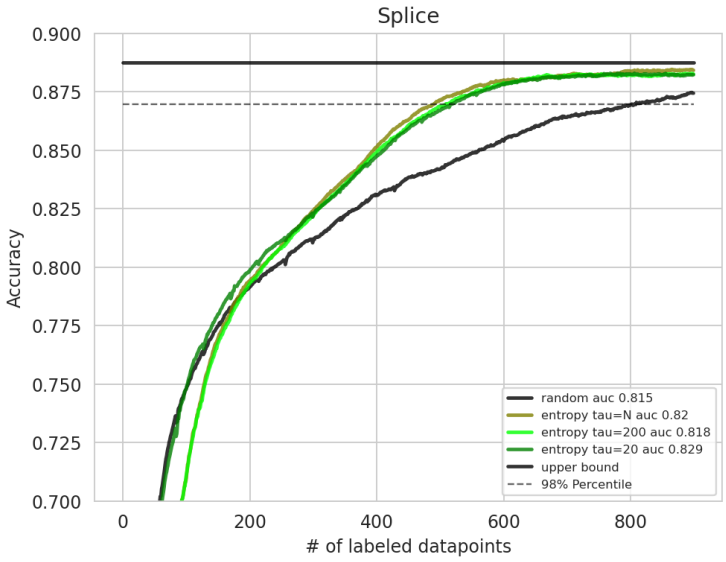
	TopV2	News
Oracle	0.860 +- 0.01	0.448 +- 0.01
SAL	0.831 +- 0.01	0.312 +- 0.01
Coreset	0.816 +- 0.02	0.355 +- 0.02
TypiClust	0.718 +- 0.02	0.314 +- 0.01
MarginScore	0.821 +- 0.02	0.357 +- 0.01
ShannonEntropy	0.794 +- 0.02	0.342 +- 0.01
RandomAgent	0.776 +- 0.02	0.349 +- 0.01
Badge	0.824 +- 0.01	0.343 +- 0.01
BALD	0.825 +- 0.02	0.347 +- 0.01

Table 6: AUC values for all algorithms on the text datasets. Higher is better.

B Alternative Plot for Restarts Ablation



C Comparison of different sample sizes



D AL Pseudocode

Algorithm 3 Active Learning

Require: $\mathcal{U}$ ▷ Unlabeled Pool
Require: τ ▷ Unlabeled Sample Size
Require: Ω ▷ AL Agent
Require: ω ▷ Environment State function
$\mathcal{L}^{(1)} \leftarrow \text{seed}(\mathcal{U})$ ▷ Create the initial labeled set
$\mathcal{U}^{(1)} \leftarrow \mathcal{U}$
for $i := 1 \dots B$ **do**
    $\text{acc}^{(i)} \leftarrow \text{Retrain}(\mathcal{L}^{(i)})$
    $a^{(i)} \leftarrow \Omega(\mathcal{U}^{(i)})\ ;\ \ a \in 1 : |\mathcal{U}^{(i)}|$
    $y^{(i)} \leftarrow \text{label}(\mathcal{U}_a^{(i)})$
    $\mathcal{L}^{(i+1)} \leftarrow \mathcal{L}^{(i)} \cup \{(\mathcal{U}_a^{(i)}, y^{(i)})\}$
    $\mathcal{U}^{(i+1)} \leftarrow \mathcal{U}^{(i)} \setminus \{\mathcal{U}_a^{(i)}\}$
end for
return $\frac{1}{B}\sum_{i=1}^{B} \text{acc}^{(i)}$