Towards Comparable Active Learning

May 5, 2023

1 Introduction

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 Contribution

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)

Manual NIPS Benchmark paper eval (out of 6):

- Has new method: 1

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

- Multiple domains: 2
- >5 datasets: 3
- >5 baselines: 4

3.1 Problem Description

Version: 1.0
Given two spaces $\mathcal{X} := \mathcal{R}^M$ and $\mathcal{Y} := \mathcal{R}^C$, a sample $\mathcal{D}_1, \ldots, \mathcal{D}_N \subseteq (\mathcal{X} \times \mathcal{Y})^*$ 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:

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

(*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 apply two relaxations to the problem. Firstly, we allow sequential construction of $\mathcal{D}_{\text{train}}$ and secondly, we limit the choice of data points in every iteration to a random sample of size τ.

To construct the active learning setting, we 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.

Framing AL as RL
We define the active learning process as an adapted reinforcement learning loop $(S, A, \tau, \Omega, \omega)$ where an environment iteratively will expose a state $s \in S$ to an agent Ω, which will choose actions $a \in A$. For each

iteration i the environment samples a subset of size τ of unlabeled instances $u^{(i)} \underset{\tau}{\sim} \mathcal{U}^{(i)}$, constructs the state $s^{(i)} := \omega(u^{(i)})$ and presents it to the agent to select an action $a^{(i)} := \Omega(s^{(i)})$. The action $a^{(i)}$ is the index of the selected instance in $u^{(i)}$ out of all possible indices $A := [1 \dots \tau]$. This process is repeated B times $i := [1 \dots B]$.

Algorithm 1 Active Learning OLD

```
Require: $\mathcal{U}$                                   ▷ Unlabeled Pool
Require: $\tau$                                   ▷ Unlabeled Sample Size
Require: $\Omega$                                   ▷ AL Agent
Require: $\omega$                                   ▷ Environment State function
 1: $\mathcal{L}^{(1)} \leftarrow \text{seed}(\mathcal{U})$                 ▷ Create the initial labeled set
 2: $\mathcal{U}^{(1)} \leftarrow \mathcal{U}$
 3: for $i := 1 \dots B$ do
 4:     $\text{acc}^{(i)} \leftarrow \text{Retrain}(\mathcal{L}^{(i)})$   ▷ $\text{Retrain}(\mathcal{L}^{(i)})$ is shorthand for $\text{Retrain}(\mathcal{L}^{(i)}, \mathcal{L}^{\text{test}}, \hat{y}_\theta, e^{\max})$
 5:     $u^{(i)} \underset{\tau}{\sim} \mathcal{U}^{(i)}$
 6:     $s^{(i)} \leftarrow \omega(u^{(i)})$
 7:     $a^{(i)} \leftarrow \Omega(s^{(i)})$                 ▷ $a^{(i)}$ is an index inside of $u^{(i)}$
 8:     $y^{(i)} \leftarrow \text{label}(u_a^{(i)})$                 ▷ $u_a^{(i)}$ is shorthand for $u_{a^{(i)}}^{(i)}$
 9:     $\mathcal{L}^{(i+1)} \leftarrow \mathcal{L}^{(i)} \cup \{(u_a^{(i)}, y^{(i)})\}$
10:     $\mathcal{U}^{(i+1)} \leftarrow \mathcal{U}^{(i)} \setminus \{u_a^{(i)}\}$
11: end for
12: return $\frac{1}{B} \sum_{i=1}^{B} \text{acc}^{(i)}$
```

Algorithm 2 Retrain

```
Require: $\mathcal{L}$                 ▷ Labeled Pool
Require: $\mathcal{L}^{\text{test}}$      ▷ Labeled Test Data
Require: $\hat{y}_\theta$          ▷ Classification Model
Require: $e^{\max}$        ▷ Maximum Epochs
 1: $\text{loss}^* \leftarrow \infty$
 2: for $i := 1 \dots e^{\max}$ do
 3:     $\theta_{i+1} \leftarrow \theta_i - \eta \nabla_\theta \ell(\mathcal{L}, \hat{y}_\theta)$
 4:     $\text{loss}_i \leftarrow \ell(\mathcal{L}^{\text{test}}, \hat{y}_\theta)$
 5:     if $\text{loss}_i < \text{loss}^*$ then
 6:         $\text{loss}^* \leftarrow \text{loss}_i$
 7:     else
 8:         Break
 9:     end if
10: end for
11: return $\text{Acc}(\mathcal{L}^{\text{test}}, \hat{y}_\theta)$
```

4 Methodology

4.1 Reproducibility

Version: 1.0

A big focus in this work is to provide an experimental setup that is fully reproducible independent from 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 sequence of batches $x_0, \dots, x_B$. Even though different AL algorithms will pick different samples from these batches, making them unavailable for sampling in later batches, the theoretical decision tree for every possible choice in every iteration i should stay the same. This behavior is not possible with the default choice of setting a global seed at the start of the experiment, since a single additional random draw from the random number generator completely changes the decision tree for the batches. This additional random number might be drawn during the initialization of the classification model, or the AL algorithm, or even during every AL iteration if Ω is stochastic. This problem also applies to the initialization of the classification models θ, the initialization and querying of $Omega$ and the drawn mini batches during the training of θ. The desired control over all these processes can be archived by assigning a separate seeded 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 on-going problem of inconsistent results of AL algorithms in different papers.

4.2 Comparibility

Version: brain dump

We split the evaluation model that generates accuracy scores and the model that generates features for the

Algorithm 3 Oracle OLD

```
Require: $\mathcal{U}$                                        ▷ Unlabeled Pool
Require: $\tau$                                               ▷ Unlabeled Sample Size
Require: $\Omega$                                             ▷ AL Agent
Require: $\omega$                                             ▷ Environment State function
 1: $\mathcal{L}^{(1)} \leftarrow \text{seed}(\mathcal{U})$   ▷ Create the initial labeled set
 2: $\mathcal{U}^{(1)} \leftarrow \mathcal{U}$
 3: for $i := 1 \ldots B$ do
 4:     $\text{acc}^{(i)} \leftarrow \text{Retrain}(\mathcal{L}^{(i)})$   ▷ $\text{Retrain}(\mathcal{L}^{(i)})$ is shorthand for $\text{Retrain}(\mathcal{L}^{(i)}, \mathcal{L}^{\text{test}}, \hat{y}_\theta, e^{\text{max}})$
 5:     $u^{(i)} \underset{\tau}{\sim} \mathcal{U}^{(i)}$
 6:     $r* \leftarrow -\infty$
 7:     $j* \leftarrow -1$
 8:     for $j := 1 \ldots \tau$ do                            ▷ Testing every unlabeled point
 9:         $y^{(j)} \leftarrow \text{label}(u_j^{(i)})$
10:         $\mathcal{L}^{(j)} \leftarrow \mathcal{L}^{(i)} \cup \{(u_j^{(i)}, y^{(j)})\}$
11:         $\text{acc}^{(j)} \leftarrow \text{Retrain}(\mathcal{L}^{(j)})$
12:         $r^{(j)} \leftarrow \text{acc}^{(j)} - \text{acc}^{(i)}$
13:         if $r^{(j)} > r^*$ then                         ▷ Select point with largest increase in performance
14:             $r* \leftarrow r^{(j)}$
15:             $j* \leftarrow j$
16:         end if
17:     end for
18:     $y^{(i)} \leftarrow \text{label}(u_{j*}^{(i)})$
19:     $\mathcal{L}^{(i+1)} \leftarrow \mathcal{L}^{(i)} \cup \{(u_{j*}^{(i)}, y^{(i)})\}$
20:     $\mathcal{U}^{(i+1)} \leftarrow \mathcal{U}^{(i)} \setminus \{u_{j*}^{(i)}\}$
21: end for
22: return $\frac{1}{B} \sum_{i=1}^{B} \text{acc}^{(i)}$
```

AL agent.
This allows for dropout / ensemble approaches to invest more compute to generate features
The normal evaluation model is available as a fallback of course
This way, the comparison is always fair

4.3 Evaluation

Version: 0.1
We compare different AL algorithms based on their archived AUC score (Eq. 1). This score does incorporate 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.
TODO

4.4 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 best sequence and used the best found solution after a fixed time budget. Even though the 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 the 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. 3.

5 Implementation Details

5.1 State Space

Version: 0.1 Remove analogies to RL
Since every AL agent needs a different state space our environment exposes a full state to the agent, so that the agent has full control of what information will be used.

The state can include the following information:

- The entire labeled dataset $\mathcal{L}^{(i)}$
- The entire unlabeled dataset $\mathcal{U}^{(i)}$
- A histogram of labeled points per class (count)
- The available budget B
- Number of added datapoints $|\mathcal{L}| - |\mathcal{S}|$

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.

- The initial validation accuracy $\text{acc}^{(0)}$ and current validation accuracy $\text{acc}^{(i)}$
- The current classification model including all model weights $\theta^{(i)}$
- The current optimizer including it's full state
- The current sample of unlabeled points $u^{(i)}$

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 4.3 we 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].

	Splice	DNA	USPS
Oracle	0.835 $\pm$ 0.01	0.879 $\pm$ 0.01	0.870 $\pm$ 0.01
SAL_23_01_16	0.808 $\pm$ 0.01	0.857 $\pm$ 0.02	0.865 $\pm$ 0.01
Coreset_Greedy	0.808 $\pm$ 0.01	0.853 $\pm$ 0.02	0.849 $\pm$ 0.01
MarginScore	0.806 $\pm$ 0.01	0.855 $\pm$ 0.02	0.864 $\pm$ 0.01
ShannonEntropy	0.808 $\pm$ 0.01	0.855 $\pm$ 0.02	0.864 $\pm$ 0.01
RandomAgent	0.797 $\pm$ 0.01	0.823 $\pm$ 0.02	0.831 $\pm$ 0.01

	Cifar10	FashionMnist
Oracle	0.718 $\pm$ 0.01	0.721 $\pm$ 0.01
SAL_23_01_16	0.660 $\pm$ 0.01	0.667 $\pm$ 0.01
Coreset_Greedy	0.668 $\pm$ 0.01	0.674 $\pm$ 0.01
MarginScore	0.674 $\pm$ 0.01	0.677 $\pm$ 0.01
ShannonEntropy	0.671 $\pm$ 0.01	0.680 $\pm$ 0.01
RandomAgent	0.632 $\pm$ 0.02	0.633 $\pm$ 0.01

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 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

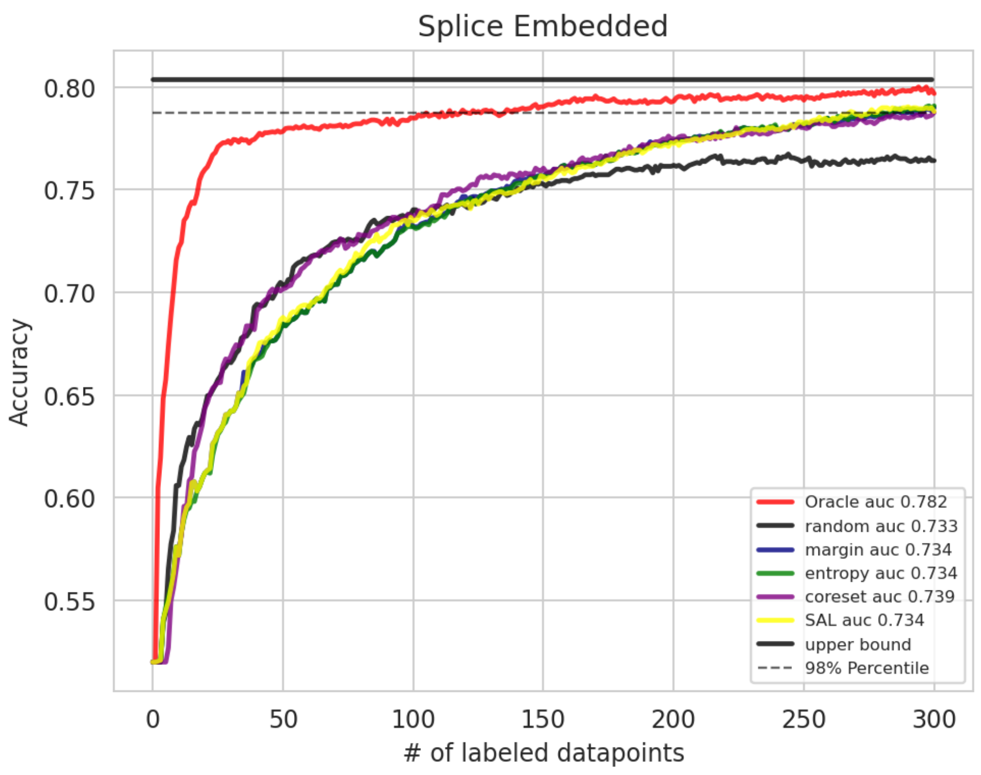


Figure 1: Results for all algorithms on the pre-encoded Splice dataset

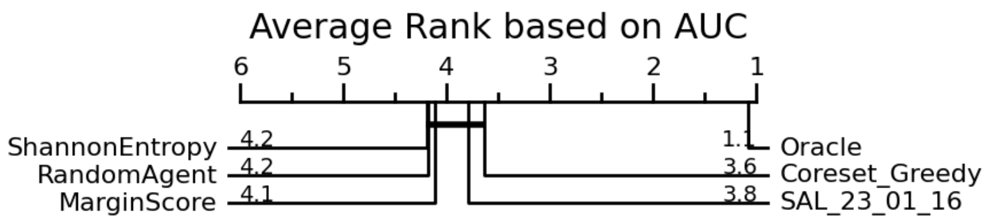


Figure 2: Critical Difference Diagram for Splice where every restart of the algorithm is one sample for the Wilcoxon-Holm method

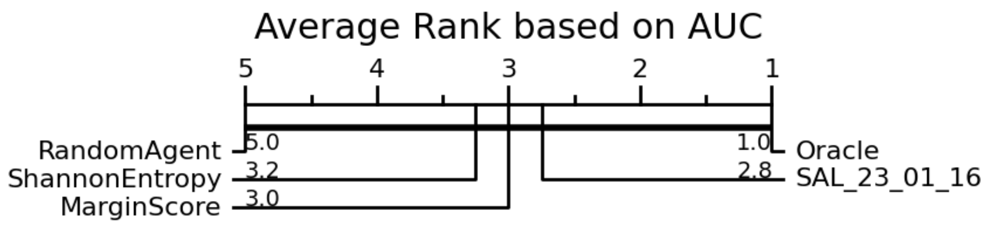


Figure 3: Critical Difference Diagram for Splice, DNA, USPS and Cifar10

7 Ablation Studies

- Setting τ to $|\mathcal{U}|$
- 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.

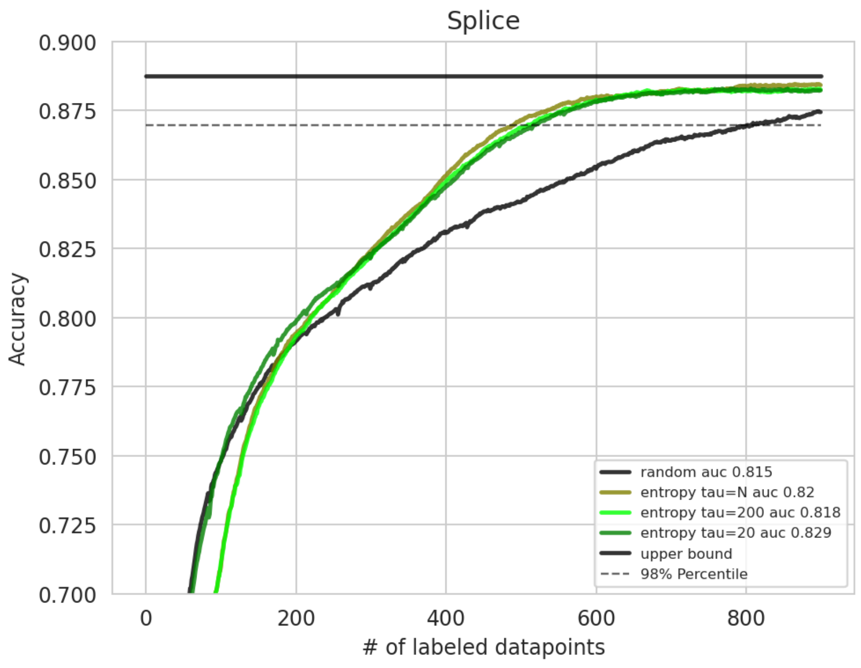
[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[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 Comparison of different sample sizes τ


Splice
Accuracy
# of labeled datapoints
random auc 0.815
entropy tau=N auc 0.82
entropy tau=200 auc 0.818
entropy tau=20 auc 0.829
upper bound
98% Percentile