
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: Braindump

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 than 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.

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

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

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

45

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 $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathcal{R}$ called loss, and $A : (\mathcal{X} \times \mathcal{Y})^* \times \mathcal{X}^* \rightarrow \mathcal{Y}^{\mathcal{X}}$ called learning algorithm, find a function

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

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

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

(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, we suppress the labels \mathcal{Y} of $\mathcal{D}_{\text{train}}$ to form the unlabeled pool $\mathcal{U} := \mathcal{X}^*$ and form an initial labeled pool by uniformly sampling k number of instances per class from \mathcal{U} and recovering their label $\mathcal{L} := (\mathcal{X}, \mathcal{Y})^{k \times C}$.

60

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

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

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

65 Framing AL as RL

66 We define the active learning process as an adapted reinforcement learning loop $(S, A, \tau, \Omega, \omega)$
 67 where an environment iteratively will expose a state $s \in S$ to an agent Ω , which will choose actions
 68 $a \in A$. For each iteration i the environment samples a subset of size τ of unlabeled instances
 69 $u^{(i)} \sim_{\tau} \mathcal{U}^{(i)}$, constructs the state $s^{(i)} := \omega(u^{(i)})$ and presents it to the agent to select an action
 70 $a^{(i)} := \Omega(s^{(i)})$. The action $a^{(i)}$ is the index of the selected instance in $u^{(i)}$ out of all possible
 71 indices $A := [1 \dots \tau]$. This process is repeated B times $i := [1 \dots B]$.

Algorithm 1 Active Learning

Require: \mathcal{U} ▷ Unlabeled Pool
Require: τ ▷ Unlabeled Sample Size
Require: Ω ▷ AL Agent
Require: ω ▷ 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: $\text{acc}^{(i)} \leftarrow \text{Retrain}(\mathcal{L}^{(i)})$
 5: $u^{(i)} \sim \text{unif}(1 : |\mathcal{U}^{(i)}|)$
 6: $a^{(i)} \leftarrow \Omega(u^{(i)}) \mid a \in 1 : |u^{(i)}|$
 7: $y^{(i)} \leftarrow \text{label}(u_a^{(i)})$
 8: $\mathcal{L}^{(i+1)} \leftarrow \mathcal{L}^{(i)} \cup \{(u_a^{(i)}, y^{(i)})\}$
 9: $\mathcal{U}^{(i+1)} \leftarrow \mathcal{U}^{(i)} \setminus \{u_a^{(i)}\}$
 10: **end for**
 11: **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}_{θ} ▷ 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** $\text{Acc}(\mathcal{L}^{\text{test}}, \hat{y}_{\theta})$

73 4 Methodology

74 4.1 Reproducibility

75 Version: 1.0

76 A big focus in this work is to provide an experimental setup that is fully reproducible independent
 77 from the dataset, classification model or AL algorithm used. In our opinion, an evaluation on one
 78 dataset with a given seed should always be done on the same sequence of batches x_0, \dots, x_B .
 79 Even though different AL algorithms will pick different samples from these batches, making them
 80 unavailable for sampling in later batches, the theoretical decision tree for every possible choice in
 81 every iteration i should stay the same. This behavior is not possible with the default choice of setting
 82 a global seed at the start of the experiment, since a single additional random draw from the random
 83 number generator completely changes the decision tree for the batches. This additional random
 84 number might be drawn during the initialization of the classification model, the AL algorithm, or
 85 even during every AL iteration if Ω is stochastic. This problem also applies to the initialization of
 86 the classification model θ , the initialization and querying of Ω and the drawn mini batches during the
 87 training of θ . The desired control over all these processes can be archived by assigning a separate
 88 seeded random number generator to all these processes. To the best of our knowledge, we are the
 89 first work that discusses this issue and proposes a solution for it. We hypothesize that the insufficient

Algorithm 3 Oracle

Require: \mathcal{U} ▷ Unlabeled Pool
Require: τ ▷ Unlabeled Sample Size
Require: Ω ▷ AL Agent
Require: ω ▷ Environment State function
▷ Create the initial labeled set
1: $\mathcal{L}^{(1)} \leftarrow \text{seed}(\mathcal{U})$
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 $\text{Retrain}(\mathcal{L}^{(i)}, \mathcal{L}^{\text{test}}, \hat{y}_\theta, e^{\text{max}})$
5: $u^{(i)} \sim \text{unif}(1 : |\mathcal{U}^{(i)}|)$
6: $r^* \leftarrow -\infty$
7: $j^* \leftarrow -1$
8: **for** $j := 1 \dots \tau$ **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** $\frac{1}{B} \sum_{i=1}^B \text{acc}^{(i)}$

90 setup with global seeds contributes to the on-going problem of inconsistent results of AL algorithms
91 in different papers.

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

96 4.2 Evaluation

97 Version: 1.0

98 We compare different AL algorithms based on their median AUC score (Eq. 1) across multiple
99 restarts of the experiment. This score does incorporate performance in early stages (low budget) as
100 well as capabilities to push the classifier in later stages (high budget). A good AL algorithm should
101 be able to perform well in both scenarios.

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

108 Additionally, we provide evidence in Fig. 1 that previous works have not evaluated their experi-
109 ments with a sufficient number of restarts. To create one slice in Fig. 1, we used all our runs for
110 the Splice dataset and drew multiple sub-sets of that respective size (X axis), effectively simulating
111 an experiment with that many restarts. Calculating the AUC values for every drawn run allows us

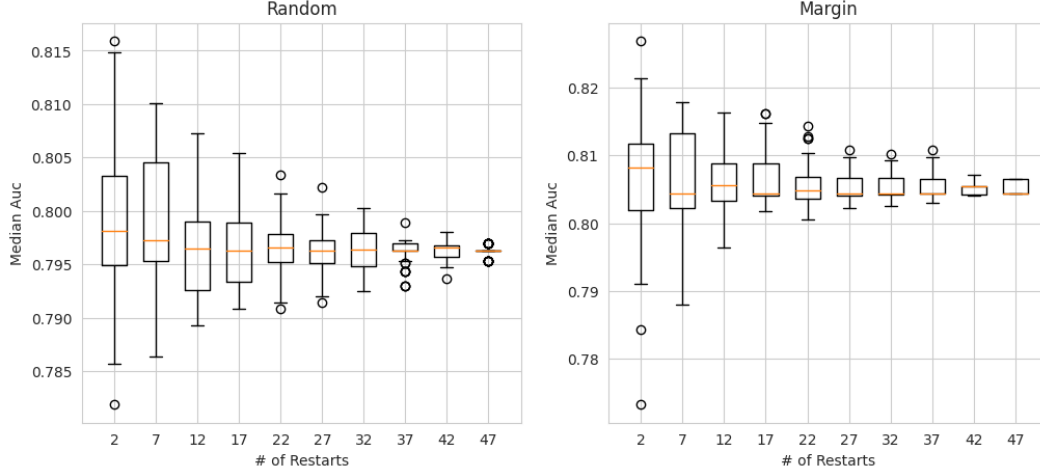


Figure 1: Random draws from Splice with different numbers of repetitions.

to draw the box-plot. Each box 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 this uncertain evaluation, we propose to repeat every experiment 50 times, to arrive at the true median AUC for each algorithm.

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. 3.

5 Implementation Details

5.1 Sampling Strategies

Uncertainty Sampling

BALD

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.

140 **Coreset**

141 **TypiClust**

142 **5.1.1 Honorable Mentions**

143 **BADGE** Relies on K-Means initialization in embedded space to pick points. If only one point is
 144 picked the most extreme outlier will be chosen. Does not make a good sampling strategy.

145 **5.2 Choosing the Classifier**

146 Traditionally, the classifier is chosen per dataset so that it is capable of solving the dataset close
 147 to the SOTA performance reported in the literature. Similar to our hypothesis in section 4.2 we
 148 hypothesize that AL algorithms will perform similarly on small classifiers and more complex ones,
 149 so that the overall ranking of algorithms stays the same. **TODO: Transform this into full hypothesis**
 150 **incl. formatting?**

151 On the basis of this hypothesis we opt to use smaller classifiers that still solve the dataset to a
 152 reasonable degree. Smaller classifiers also require fewer labeled datapoints to acquire performance
 153 close to the upper bound. This behavior can be observed in Figure **TODO**

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

157 **5.3 Training the Classifier**

158 **Version: 0.1 Full rework needed**

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

167 **6 Experiments**

168 **6.1 Datasets**

169 **Version: 1.0**

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

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

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

173 **Tabular** We use **Splice**, **DNA** and **USPS** from LibSVMTools [10]. All three datasets are normal-
174 ized between [0, 1].

175 **Image** We use **FashionMNIST** [13] and **Cifar10** [7]. Both datasets are normalized between [-1,
176 1].

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

182 We would like to point out that these datasets can be considered "toy-datasets" and therefore not
183 relevant for practical purposes. This might be true if we aimed to develop novel classification models
184 on these datasets, however we are solely focused on comparing different AL algorithms in this paper.
185 Our core assumption is that a well-performing algorithm in our benchmark will transfer well into
186 more practical use-cases.

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

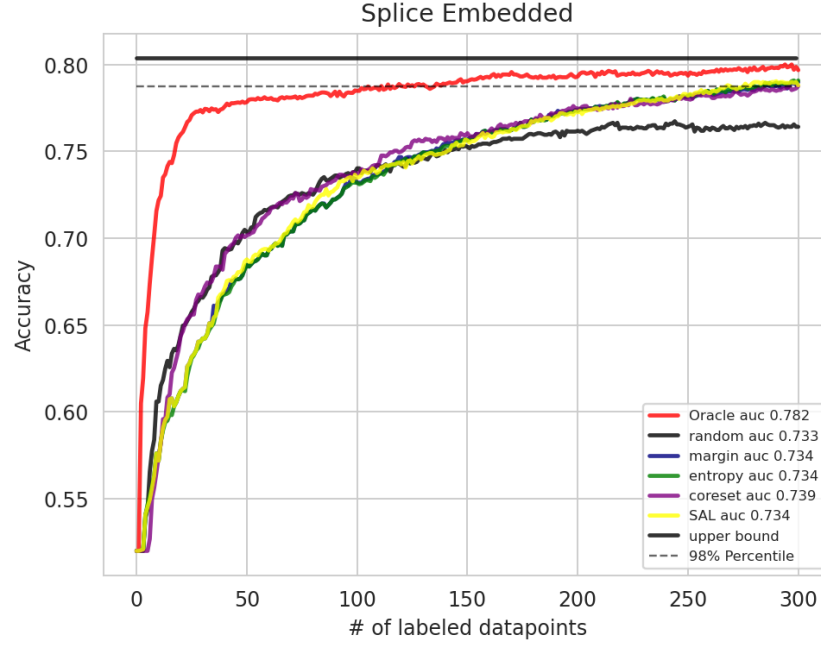


Figure 2: Results for all algorithms on the pre-encoded Splice dataset

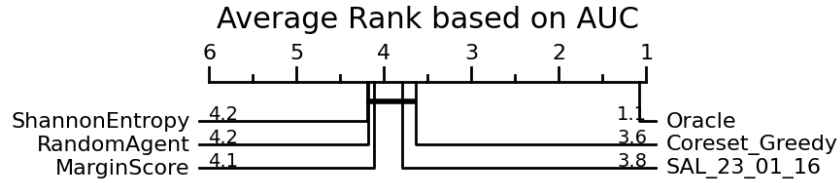


Figure 3: Critical Difference Diagram for Splice where every restart of the algorithm is one sample for the Wilcoxon-Holm method

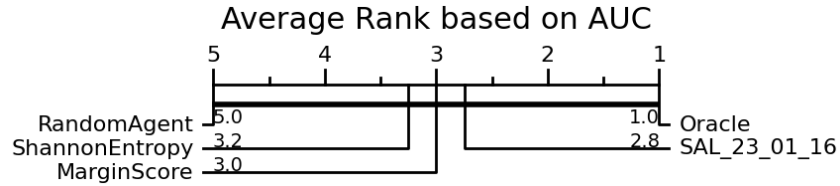


Figure 4: Critical Difference Diagram for Splice, DNA, USPS and Cifar10

195 **7 Ablation Studies**

- 196 • Setting τ to $|\mathcal{U}|$
- 197 • 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.

234 **A Comparison of different sample sizes**

