Towards Understanding the Optimal Behaviors of Deep Active Learning Algorithms

Yilun Zhou* **Adi Renduchintala†** **Xian Li†**
Sida Wang† **Yashar Mehdad†** **Asish Ghoshal†**

Abstract

Active learning (AL) algorithms may achieve better performance with fewer data because the model guides the data selection process. While many algorithms have been proposed, there is little study on what the *optimal* AL algorithm looks like, which would help researchers understand where their models fall short and iterate on the design. In this paper, we present a simulated annealing algorithm to search for this optimal oracle and analyze it for several different tasks. We present several qualitative and quantitative insights into the optimal behavior and contrast this behavior with those of various heuristics. When augmented by with one particular insight, heuristics perform consistently better. We hope that our findings can better inform future active learning research. The code for the experiments is available at `https://github.com/YilunZhou/optimal-active-learning`.

1 Introduction

Training deep models typically requires a large dataset, which limits its usefulness in domains where expensive expert annotation is required. Traditionally, active learning (AL) (Settles, 2009), in which the model can select the data to learn from, is often presented as a more sample efficient alternative to standard supervised learning. However the gain of AL with deep models is less consistent. For example, the best performing method seems to depend on the task in an unpredictable manner (Lowell et al., 2018).

Is active learning still useful in the era of deep learning? Although many issues have been found, it is not clear whether those problems only plague current AL methods or also apply to methods developed *in the future*. Moreover, in existing literature, there lacks a comparison of proposed methods to the *oracle upper-bound*. Such a comparison is helpful for debugging machine learning models. For example, the classification confusion matrix may reveal the inability of the model to learn a particular class, and a comparison between a sub-optimal RL policy and the human expert play may indicate a lack of exploration. By contrast, without such an oracle reference in AL, it is extremely difficult, if at all possible, to pinpoint the inadequacy of an AL method and improve it.

In this paper, we propose a simulated annealing algorithm to search for the *optimal* AL strategy for a given base learner. With practical computational resources, this procedure is able to find an oracle that significantly outperforms existing heuristics (average improvement of +7.53% from random order, compared to +1.49% attained by the best heuristics across three vision and language tasks), for models both trained from scratch and finetuned from pre-training, ***definitively asserting the usefulness of a high-performing AL algorithm in most scenarios.*** We also present the following insights into the oracle behavior.

While many papers do not explicitly state whether and how training stochasticity (e.g. model initialization or dropout) is controlled across iterations, we show that ***training stochasticity tends to negatively affect the oracle performance (Sec. 6.2).***

Previous work (Lowell et al., 2018) has found that for several heuristic methods, the actively acquired dataset does not transfer across different architectures. We observed a lesser extent of this phenomenon, but more importantly, ***the oracle transfers better than heuristics (Sec. 6.3).***

It may seem reasonable that a high-performing AL algorithm should exhibit a non-uniform sampling behavior (e.g. focusing more on harder to learn regions). However, ***the oracle mostly preserves data distributional properties (Sec. 6.4).***

Finally, using the previous insight, ***heuristics can on average be improved by 2.95% with a simple distribution-matching regularization (Sec. 6.5)***.

* MIT CSAIL. Work done during an internship at Facebook AI. Correspondence to Yilun Zhou: `yilun@mit.edu`.
† Facebook AI.

2 Related Work

Active learning (Settles, 2009) has been studied for a long time. At the core of an active learning algorithm is the acquisition function, which generates or selects new data points to label at each iteration. Several different types of heuristics have been proposed, such as those based on uncertainty (Kapoor et al., 2007), disagreement (Houlsby et al., 2011), diversity (Xu et al., 2007), and expected error reduction (Roy and McCallum, 2001). In addition, several recent studies focused on meta-active learning, i.e. learning a good acquisition function on a source task and transfer it to a target task (Fang et al., 2017; Woodward and Finn, 2017; Bachman et al., 2017; Konyushkova et al., 2017; Pang et al., 2018; Contardo et al., 2017; Konyushkova et al., 2018; Liu et al., 2018; Vu et al., 2019).

With neural network models, active learning has been applied to computer vision (CV) (e.g. Gal et al., 2017; Kirsch et al., 2019; Sinha et al., 2019) and natural language processing (NLP) (e.g. Shen et al., 2017; Fang et al., 2017; Liu et al., 2018; Kasai et al., 2019) tasks. However, across different tasks, it appears that the relative performance of different methods can vary widely: a method can be the best on one task while struggling to even outperform the random baseline on another, with little explanation given. We compile a meta-analysis in App. A. Moreover, Lowell et al. (2018) found that the data acquisition order does not transfer well across architectures, a particularly important issue during deployment when it is expected that the acquired dataset will be used for future model development (i.e. datasets outliving models).

The closest work to ours is done by Koshorek et al. (2019), who studied the active learning limit. There are numerous differences. First, we analyze several CV and NLP tasks, while they focused on semantic role labeling (SRL) in NLP. Second, we explicitly account for training stochasticity, shown to be important in Sec. 6.2, but they ignored it. Third, our global simulated annealing search is able to find significant gaps between the upper limit and existing heuristics while their local beam search failed to find such a gap (though on a different task). We additionally show how to improve upon heuristics with our insights.

3 Problem Formulation

3.1 Active Learning Loop

Let the input and output space be denoted by $\mathcal{X}$ and $\mathcal{Y}$ respectively, where $\mathcal{Y}$ is a finite set. Let $\mathbb{P}_{XY}$ be the data distribution over $\mathcal{X} \times \mathcal{Y}$. We consider multi-class classification where a model $m_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by θ, maps an input to an output.

We study pool-based batch-mode active learning where we assume access to an unlabeled data pool $\mathcal{D}^U$ drawn from the marginal distribution $\mathbb{P}_X$ over $\mathcal{X}$.

Starting with a labeled warm-start set $\mathcal{D}_0^L \sim \mathbb{P}_{XY}$, an AL loop builds a sequence of models $(\theta_0, \ldots, \theta_K)$ and a sequence of labeled datasets $(\mathcal{D}_1^L, \ldots, \mathcal{D}_K^L)$ in an interleaving manner. At the k-th iteration, for $k = 0, \ldots, K$, a trainer η takes in $\mathcal{D}_k^L$ and produces the model parameters θ_k by minimizing some loss function on $\mathcal{D}_k^L$. In deep-learning the model training is typically stochastic (due to, for example, random initialization and drop-out masking) and θ_k is a random variable. We assume that all such stochasticity are captured in ξ such that $\theta_k = \eta(\mathcal{D}_k^L, \xi)$ is deterministic.

Using the trained model m_{θ_k} and the current labeled set $\mathcal{D}_k^L$, an acquisition function $\mathcal{A}$ builds the dataset $\mathcal{D}_{k+1}^L$ by selecting a batch of B data points from the unlabeled pool, $\Delta\mathcal{D}_{k+1} \subseteq \mathcal{D}^U$, querying the annotator for labels, and adding them to $\mathcal{D}_k^L$; i.e. $\Delta\mathcal{D}_{k+1} = \mathcal{A}(\theta_k, \mathcal{D}_k^L); \mathcal{D}_{k+1}^L = \mathcal{D}_k^L \cup \Delta\mathcal{D}_{k+1}$.[1] The model training and data acquisition loop repeats until we obtain $\mathcal{D}_K^L$ by querying labels for KB data points, followed by training m_{θ_K} for a final time.

3.2 Performance Curve

Given a specific draw of ξ, we measure the performance of an acquisition function $\mathcal{A}$ by its *performance curve* $\tau_{\mathcal{A},\xi} : \{1, \ldots, K\} \to [0, 1]$ defined below:

$$\tau_{\mathcal{A},\xi}(k) = \mathbb{E}_{x,y\sim\mathbb{P}_{XY}}\left[e(m_{\theta_k}(x), y)\right],$$

where $e : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ is the evaluation metric (e.g. accuracy). $\theta_k = \eta(\mathcal{D}_k^L, \xi)$, and $\mathcal{D}_k^L = \mathcal{D}_{k-1}^L \cup \mathcal{A}(\theta_{k-1}, \mathcal{D}_{k-1}^L)$.

Definition 1 (Anytime optimality)**.** *An acquisition function $\mathcal{A}$ is ξ-anytime optimal if it uniformly dominates every other acquisition function $\mathcal{A}'$ as measured by $\tau_{\cdot,\xi}$; i.e., $\tau_{\mathcal{A},\xi}(k) \geq \tau_{\mathcal{A}',\xi}(k)$ $\forall k \in \{1, \ldots, K\}$ and $\forall \mathcal{A}' \neq \mathcal{A}$.*

Proposition 1. *There exist data distribution $\mathbb{P}_{XY}$ and model class m_θ for which an anytime optimal acquisition function does not exist.*

Proof Sketch. Fig. 1 shows a counter-example. In Fig. 1(a), we have an underlying distribution $\mathbb{P}_{XY}$ shown as the colored background, and four points drawn from the distribution. If we learn a max-margin linear classifier from the four points, we can uncover the ground-truth decision boundary. If we acquire two data points, the optimal combination is shown in Fig. 1(b), resulting in a slightly wrong decision boundary. Choosing a different blue point would result in a much worse decision boundary (Fig. 1(c)). However,

[1] Some acquisition functions such as BALD are stochastic. For simplicity, we discuss the case of deterministic $\mathcal{A}$ here. Extension of our claims to the stochastic case is straightforward and presented in App. B.

the optimal three-point acquisition (Fig. 1(d)), which leads to the ground-truth decision boundary, does *not* contain the previously optimal blue point in Fig. 1(b). Thus, there is no acquisition function simultaneously optimal at both two and three data points. □

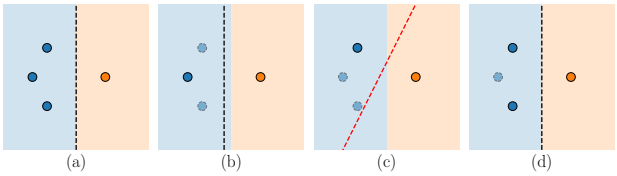

Figure 1: The counter example to prove Prop. 1.

Remark. *In most AL papers, the authors demonstrate the superiority of their proposed method by showing its performance curve visually "above" those of the baselines. However, the above result shows that such an anytime optimal acquisition function may not exist.*

3.3 ξ-Quality

To make different acquisition functions comparable, we propose the following average performance metric, which we will refer to as ξ-*quality*:

$$q_\xi(\mathcal{A}) \stackrel{\text{def}}{=} \frac{1}{K}\sum_{k=1}^{K} \tau_{\mathcal{A},\xi}(k).$$

We refer to the acquisition strategy that maximizes this ξ-quality as ξ-*optimal* strategy, denoted as $\mathcal{A}^*_\xi$. Obviously, the ξ-anytime optimal acquisition function, if exists, will also be ξ-optimal.

While we could define the quality simply as the end point performance, $\tau_{\mathcal{A},\xi}(K)$, doing so fails to distinguish acquisition functions that lead to a fast initial performance rise from the rest, potentially incurring more annotation cost than necessary.

There are two interpretations to the above quality definition. First, it is the right Riemann sum approximation of the area under curve (AUC) of τ_a, which correlates with intuitive notions of optimality for acquisition functions. Second, the un-averaged version of our definition can also be interpreted as the undiscounted cumulative reward of an acquisition *policy* over K steps where the per-step reward is the resulting model performance following data acquisition. Thus, the optimal policy implicitly trades off between immediate ($k = 1$) and future ($k \to K$) model performances.

3.4 Optimal Data Labeling Order

We index data points in $\mathcal{D}^U$ by $\mathcal{D}^U_i$ and use $y\left(\mathcal{D}^U_i\right)$ to denote the label. Since an acquisition function generates a sequence of labeled datasets $\mathcal{D}^L_1 \subset \ldots \subset \mathcal{D}^L_K$, an acquisition function is equivalent to a partial permutation order σ of KB indices of $\mathcal{D}^U$ with $\mathcal{D}^L_k = \mathcal{D}^L_0 \cup \{(\mathcal{D}^U_{\sigma_i}, y\left(\mathcal{D}^U_{\sigma_i}\right))\}_{i\in[kB]}$. Thus, the problem of searching for $\mathcal{A}^*_\xi$ reduces to that of searching for the ξ-optimal order σ^*_ξ, which also relieves us from explicitly considering different forms of acquisition functions.

Moreover, a direct implication of above definitions is that optimal order depends on ξ; i.e. $q_\xi(\sigma^*_\xi) \geq q_\xi(\sigma^*_{\xi'})$, for $\xi \neq \xi'$. As we experimentally demonstrate in Sec. 6.2, such a gap does exist. Since stochasticity in model training is completely controllable, an acquisition function that approaches optimal limit may need to explicitly take such randomness into account.

4 Search for the Optimal Order

There are two technical challenges in finding σ^*_ξ. First, we do not have access to the data distribution. Second, the permutation space of all orders is too large.

4.1 Validation Set Maximum

To solve the first problem, we assume access to a validation set $\mathcal{D}^V \sim \mathbb{P}_{XY}$. Since we are searching for the oracle model, there is no practical constraints on the size of $\mathcal{D}^V$. In addition, we assume access to an independently drawn test set $\mathcal{D}^T \sim \mathbb{P}_{XY}$. We define our optimal order estimator as

$$\widehat{\sigma}_\xi = \arg\max_\sigma q_\xi\left(\sigma, \mathcal{D}^V\right), \tag{1}$$

and its quality on the test set $q_\xi\left(\sigma, \mathcal{D}^T\right)$ serves as the an *unbiased* estimate of its generalization quality.

4.2 Simulated Annealing Search

Exhaustively searching over the space of all labeling orders is prohibitive as there are ${}^{|\mathcal{D}^U|}P_{BK} = |\mathcal{D}^U|!/(BK)!$ different orders to consider. In addition, we cannot use gradient-based optimization due to the discreteness of order space. Thus, we propose a simulated annealing (SA)-based approach, shown in Alg. 1.

The search starts with a randomly initialized order $\sigma^{(0)}$. At time step t, it proposes a new order $\sigma^{(p)}$ from $\sigma^{(t-1)}$ with the following transition kernel (Fig. 2): with equal probabilities, it either swaps two data points from two different batches or replaces a data point in use by an unused one. It then evaluates the quality of the new order and accepts or rejects the proposal with probability depending on the quality difference. After searching for T_S steps, the algorithm greedily optimizes the best order for an additional T_G steps, and returns best order and quality in the end.

5 Experiment Overview

In addition to datasets $\mathcal{D}^U, \mathcal{D}^L_0, \mathcal{D}^V$ and $\mathcal{D}^T$ described in Sec. 3 and 4, since training neural networks on small datasets is prone to overfitting, we introduced an additional model selection set $\mathcal{D}^M$ to select the best model

Task	Type	Dataset	$\lvert\mathcal{D}^U\rvert, \lvert\mathcal{D}_0^L\rvert, \lvert\mathcal{D}^M\rvert, \lvert\mathcal{D}^V\rvert, \lvert\mathcal{D}^T\rvert$	$B,\ K$	Metric	Architecture	Heuristics
Object Cls.	class.	Fashion-MNIST	2000, 50, 150, 4000, 4000	25, 12	Acc	CNN	Max-Ent., (Batch)BALD
Intent Cls.	class.	TOPv2 (alarm)	800, 40, 100, 4000, 4000	20, 8	F1	LSTM, CNN, AOE, RoBERTa	Max-Ent., BALD
NER	tagging	MIT Restaurant	1000, 50, 200, 3000, 3000	25, 10	F1	LSTM	(Norm.-)Min-Conf., Longest

Table 1: Summary of experiment settings. Architecture details are in App. C.

Algorithm 1: Simulated Annealing (SA) Search

Input: SA steps T_S, greedy steps T_G, linear annealing factor α

$\sigma^{(0)} = \texttt{random-shuffle}\left([1, 2, ..., |\mathcal{D}^U|]\right)$;
$q^{(0)} = q_\xi(\sigma^{(0)})$;
$\sigma^* = \sigma^{(0)}; q^* = q^{(0)}$;
for $t = 1, 2, ..., T_S$ **do**
  $\sigma^{(p)} = \texttt{propose}\left(\sigma^{(t-1)}\right)$;
  $q^{(p)} = q_\xi\left(\sigma^{(p)}, \mathcal{D}^V\right)$;
  $u \sim \text{Unif}[0, 1]$;
  if $u < \exp\left[\alpha t \left(q^{(p)} - q^{(t-1)}\right)\right]$ **then**
    $\sigma^{(t)} = \sigma^{(p)}; q^{(t)} = q^{(p)}$;
    if $q^* < q^{(p)}$ **then**
      $\sigma^* = \sigma^{(p)}; q^* = q^{(p)}$;
  else
    $\sigma^{(t)} = \sigma^{(t-1)}; q^{(t)} = q^{(t-1)}$;
for $t = 1, 2, ..., T_G$ **do**
  $\sigma^{(p)} = \texttt{propose}\left(\sigma^*\right)$;
  $q^{(p)} = q_\xi\left(\sigma^{(p)}, \mathcal{D}^V\right)$;
  if $q^{(p)} > q^*$ **then**
    $\sigma^* = \sigma^{(p)}; q^* = q^{(p)}$;
return σ^*, q^*

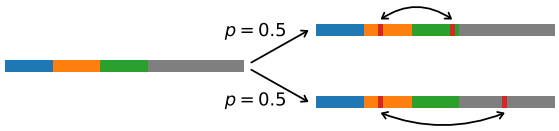

Figure 2: Illustration of the transition kernel. With equal probabilities, the kernel either swaps two data points from two different batches (sections of different colors) or replaces a data point in the first K batches with one outside (the gray section).

across 100 epochs and also trigger early stopping if the performance on this set is not improved for 20 consecutive epochs. In typical AL settings where data annotation is costly, D^M is typically comparable in size to the warm-start set $\mathcal{D}_0^L$.

For each task, Tab. 1 describes the experiment-specific parameters. The training used the batch size equal to the acquisition batch size B. Optimization used the Adam optimizer (Kingma and Ba, 2014). We searched for $T_S = 25,000$ and $T_G = 5000$ steps with $\alpha = 0.1$. We found little improvement in the greedy stage.

The full dataset is shuffled and split into non-overlapping sets $\mathcal{D}^U, \mathcal{D}_0^L, \mathcal{D}^M, \mathcal{D}^V, \mathcal{D}^T$. Since the shuffling is *not* label-stratified, the empirical label distribution in any set (and $\mathcal{D}_0^L$ in particular) may not be close to the actual distribution. We made this choice as it better reflects the actual deployment. For each task we considered commonly used architectures.

5.1 Object Classification

For object classification, we used the `Fashion-MNIST` dataset (Xiao et al., 2017) with ten classes. Even though the dataset is label-balanced, $|\mathcal{D}_0^L| = 50$ means that the initial warm-start set can be extremely imbalanced, posing additional challenge to uncertainty estimation used by many heuristics. We used a CNN architecture with two convolution layers followed by two fully connected layers. We compared against max-entropy, BALD (Gal et al., 2017), and BatchBALD (Kirsch et al., 2019) heuristics[2].

5.2 Intent Classification

For intent classification, we used the Task-Oriented Parsing v2 (`TOPv2`) dataset (Chen et al., 2020), which consists of eight domains of human interaction with digital assistants, such as *weather* and *navigation*. We used the "alarm" domain for our experiments. In intent classification, given an user instruction such as "set an alarm at 8 am tomorrow", the task is to predict the main intent of the sentence (`create-alarm`). There are seven intent classes: `create-alarm`, `get-alarm`, `delete-alarm`, `silence-alarm`, `snooze-alarm`, `update-alarm`, and `other`. The dataset is very imbalanced, with `create-alarm` accounting for over half of all examples. Hence, we used multi-class weighted F1 score as the evaluation metric.

Our main architecture of study is the Bi-directional LSTM (BiLSTM) architecture with word embeddings initialized to GloVe (Pennington et al., 2014). To study the model transfer quality (Sec. 6.3), we also considered a CNN architecture, which uses 1D convolution layers to process a sentence represented as its word embedding sequence, an Average-of-Embedding

[2](Batch)BALD implementations provided by the authors: `https://github.com/BlackHC/batchbald_redux/`.

(AOE) architecture, which uses a fully connected network to process a sentence represented by the average embedding of the words, and finetuning from the pretrained RoBERTa model (Liu et al., 2019). Detailed architectures are described in App. C. We considered max-entropy and BALD heuristics.

5.3 Named Entity Recognition

Named entity recognition (NER) is a structured prediction NLP task to predict a tag type for each word in the input sentence. We used the `MIT Restaurant` dataset (Liu et al., 2013), which consists of restaurant-related sentences tagged in Beginning-Inner-Outer (BIO) format. Tags include `amenity`, `price`, `location`, etc. For example, the tags for "*what restaurant near here serves pancakes at 6 am*" are `[O, O, B-location, I-location, O, B-dish, O, B-hours, I-hours]`. The outer tag accounts for more than half of all tags, and tag-level weighted F1 score is used as the evaluation metric.

We used a BiLSTM encoder and an LSTM decoder following Shen et al. (2017) but without the CNN character encoder as an ablation study does not find it beneficial. We used teacher-forcing at training time and greedy decoding at prediction time. During active acquisition, we assumed that the annotation cost is the same for each sentence regardless of its length. We considered several heuristics. The min-confidence heuristic selects the sentence with the lowest log joint probability of the greedily decoded tag sequence (i.e. sum of log probability of each tag), which is divided by sentence length in its normalized version. The longest heuristic prioritizes longer sentences.

6 Results and Analyses

We compared the performance of the oracle order against existing heuristics. We called the oracle order and performance "optimal" even though they are estimates of the truly optimal ones due to our use of the validation set for estimating the quality and the simulated annealing search.

6.1 Optimal Quality Values and Curves

Tab. 2 presents the qualities of the optimal, heuristic and random orders. On all three tasks, there are large gaps between the heuristic and optimal orders. Specifically, the optimal order improve upon the best heuristic by 11.7%, 3.1%, and 3.5% for object classification, intent classification, and NER respectively in terms of ξ-quality. Comparing to the random order, the optimal order improves upon it by an average of 7.53% across three tasks, while the best heuristic order only achieves an average improvement of 1.49%.

Visually, Fig. 3 depicts the performance curve of the optimal order against heuristic and random baselines on both the validation ($\mathcal{D}^V$) and test ($\mathcal{D}^T$) set for the object classification task. The optimal order significantly outperforms heuristic and random baselines, both numerically and visually, and we observed that the optimal order found using the validation set generalizes to the test set. Quality summary for each individual heuristic and performance curves for intent classification and NER tasks are presented in App. D and E. Even though there might not exist an anytime optimal acquisition function (Prop. 1), we did observe that the oracle order is uniformly better than heuristics on all three tasks.

	Object Cls.	Intent Cls.	NER
Optimal	**0.761**	**0.887**	**0.839**
Best Heuristic	0.682*	0.860	0.811
Random	0.698*	0.816	0.800

Table 2: Qualities of optimal, heuristic, and random orders on the three tasks. Individual heuristic performances are shown in App. D. *No heuristics outperform the random baseline on object classification.

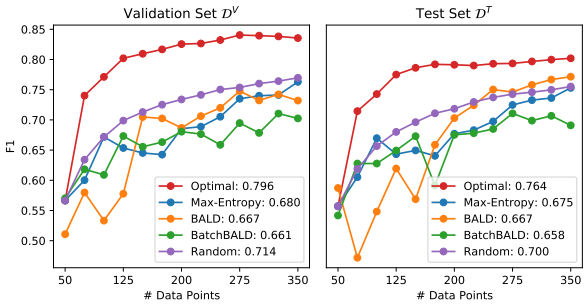

Figure 3: Performance curves of optimal, heuristic, and random orders for object classification.

6.2 Effect of Training Stochasticity

As previously stated, training stochasticity could negatively affect the optimal quality. To experimentally verify this, for ξ and ξ' we compared $q_\xi(\widehat{\sigma}_\xi, \mathcal{D}^T)$ and $q_\xi(\widehat{\sigma}_{\xi'}, \mathcal{D}^T)$ to study whether ξ'-optimal order is less optimal for ξ-training. Since we did not have dropout layers in our network, the only source of stochasticity comes from the random initialization.

For object and intent classification tasks, we considered five different seeds, $\xi, \xi' \in \{0, ..., 4\}$. Fig. 4 shows the pairwise quality $q_\xi(\widehat{\sigma}_{\xi'}, \mathcal{D}^T)$, where ξ' is the source seed on the column and ξ is the target seed on the row. The color on each cell represents the seed-mismatch quality gap $q_\xi(\widehat{\sigma}_\xi, \mathcal{D}^T) - q_\xi(\widehat{\sigma}_{\xi'}, \mathcal{D}^T)$, with a darker color indicating a larger gap.

The results suggest that in order to fully realize the

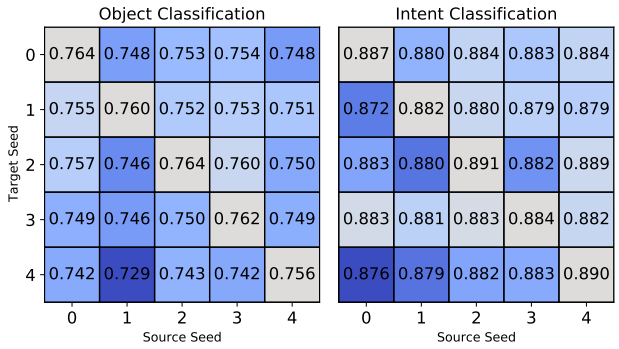

Figure 4: Training stochasticity negatively affects the optimal quality. The number in each cell represents $q_\xi(\widehat{\sigma}_{\xi'})$, with ξ on the row and ξ' on the column. The color represents $q_\xi(\widehat{\sigma}_{\xi'}) - q_\xi(\widehat{\sigma}_\xi)$, with darker colors indicating larger gaps.

potential of active learning, the acquisition function needs to specifically consider the particular model. However, high-quality uncertainty estimation are most commonly defined for a *model class* (e.g. the class of all random initialization in Deep Ensemble (Lakshminarayanan et al., 2017), the class of Dropout masks in MC-Dropout (Gal and Ghahramani, 2016), and the class of Gaussian noises in Bayes-by-Backprop (Blundell et al., 2015)). A definition and algorithms for single-model uncertainty have the potential of further improving uncertainty-based heuristics. In addition, the result also implies that a purely diversity-based method relying on pre-defined embedding or distance metric could not be optimal.

6.3 Model Transfer Quality

Lowell et al. (2018) found that the heuristic acquisition order on one model sometimes does not transfer well to another model. However, it is not clear whether this is a general bane of active learning or just about specific heuristics. On the one hand, if the optimal order finds generally valuable examples early on, we would expect a different model architecture to also benefit from this order. On the other hand, if the optimal order exploits particularities of the model architecture, we would expect it to have worse transfer quality.

We replicated this study on the intent classification task, using four very different deep architectures: BiLSTM, CNN, AOE and RoBERTa described in Sec. 5. For a pair of source-target architectures (A_s, A_t), we found both the optimal and best heuristic order on A_s and evaluate it on A_t. Then we compared its quality to that of a random order on A_t. The best heuristic is BALD for LSTM, CNN and RoBERTa, and max-entropy for AOE. Fig. 5 shows the quality of each order. The optimal order always transfers better than the random order and, with the single exception of RoBERTa → AOE, better than or equal to the heuristic order as well. This suggests that the optimal order tends to label model-agnostically valuable data points, and it is likely that better acquisition functions for one architecture can also help assemble higher-quality datasets for other architectures.

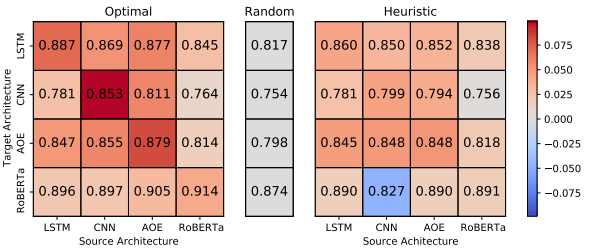

Figure 5: Quality of optimal and max-entropy heuristic order across different architectures. Cell color represents difference to the random baseline.

Perhaps surprisingly, we found that the optimal orders for different architectures actually share fewer data points than the heuristic orders, despite achieving higher transfer performance. We describe this phenomenon and present our analysis in App. F.

6.4 Distributional Characteristics

In this set of experiments, we visualized the distributional characteristics of the acquired data points under the optimal and heuristic orders. We analyzed both the input and the output space.

6.4.1 Object Classification

The top panel of Fig. 6 visualizes the first five batches (125 data points in total) acquired by the optimal and max-entropy orders. For 28×28 grayscale Fashion-MNIST images, we reduced their dimension to 100 using principal component analysis (PCA), then to 2 using t-distributed stochastic neighbor embedding (t-SNE) (Maaten and Hinton, 2008). Each acquired image, positioned at its 2-dimensional embedding and color-coded by label, is represented by a number referring to the batch index. Circle and triangle marks represent points in the test and warm-start sets.

The bottom panel of Fig. 6 visualizes label distribution of the $\mathcal{D}_k^L$ during acquisition. The horizontal axis represents the total number of data points (starting from 50 images in $\mathcal{D}_0^L$, each order acquires 300 from the pool), while the relative length of each color represents the frequency of each label. The bars between plots show the "reference distribution" from the test set, which is balanced in this task.

For both input and output, the optimal order samples quite uniformly in the sample space, and is not easily

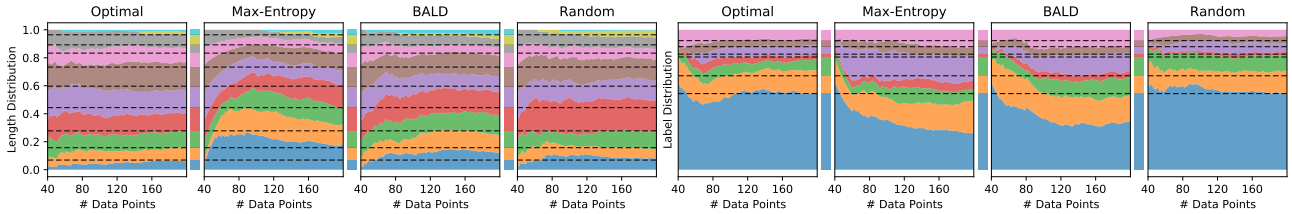

Figure 7: Sentence length and label distribution w.r.t. labeled set size for intent classification. For sentence lengths, the colors represent, from bottom to top: 1–3, 4, 5, 6, 7, 8, 9, 10, 11–12, and 13+. For labels, the colors represent, from bottom to top: `create-alarm`, `get-alarm`, `delete-alarm`, `other`, `silence-alarm`, `snooze-alarm`, and `update-alarm`.

distinguishable from random sampling. However, this order is distinctively non-random because it achieves a much higher quality than the random order (Tab. 2). By contrast, the max-entropy order is very imbalanced manner in both input and output space. BALD and BatchBALD exhibit similar behaviors (App. G).

6.4.2 Intent Classification

Fig. 7 presents the analogous analysis for the intent classification task, where we studied the sentence-length aspect of the input distribution. Since there are few sentences with less than four or more than ten words, we grouped those sentences into 0-3, 11-12, and 13+ categories for ease of visualization. Both input and output distribution plots echo those in Fig. 6.

In particular, even though short sentences (blue and orange) are infrequent in the actual data distribution, both heuristics over-samples them, likely because shorter sentences provide less "evidence" for a prediction. In addition, they also significantly under-samples the `create-alarm` label (blue), likely because it is already well-represented in the warm-start set, and over-samples the `silence-alarm` label (purple), for the converse reason. The optimal order, again, does not exhibit any of these under- or over-sampling behaviors while achieving a much higher quality.

6.4.3 Named Entity Recognition

Fig. 8 presents the analysis for the NER task. We observed the save distribution-matching property, on both the input and the output side. By contrast, even if the acquisition function does not have the "freedom" to choose sentences with arbitrary tags, the min-confidence heuristic still over-samples the relatively infrequent `B/I-hours` tags (purple) in the beginning.

Notably, in NER the basic "supervision unit" is an individual word-tag pair. Thus, the model can effectively learn from more data by selecting longer sen-

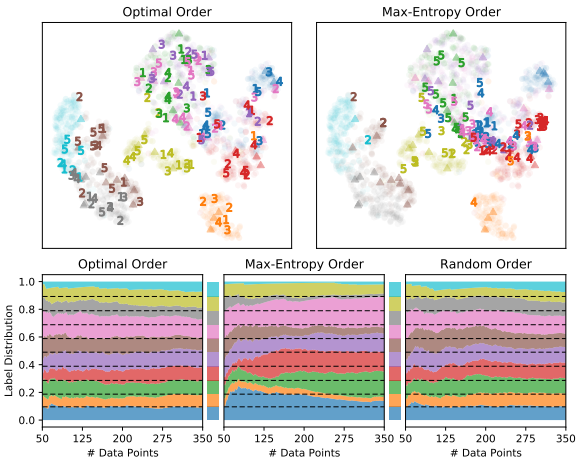

Figure 6: Distribution visualization for object classification. Top: the first five batches numerically labeled in t-SNE embedding space, along with warm-start (triangle) and test (circle) samples. Bottom: label distribution w.r.t. labeled set size, with test-set distribution shown between plots and as dashed lines.

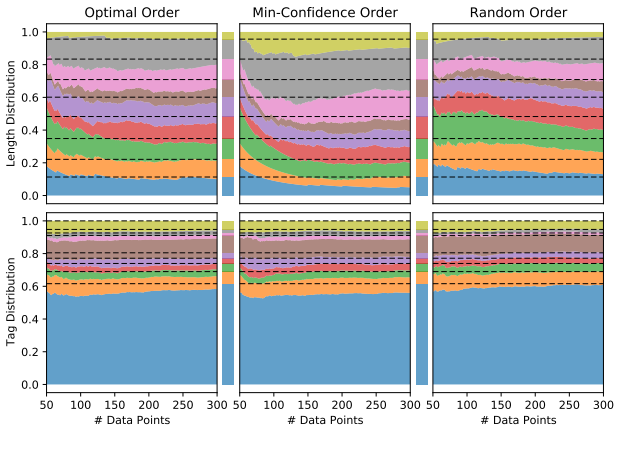

Figure 8: Sentence length and label distribution w.r.t. labeled set size for NER. For sentence lengths, the colors represent, from bottom to top: 1–5, 6, 7, 8, 9, 10, 11–12, 13-16, and 17+. For labels, the colors represent, from bottom to top: `Outer`, `B/I-amenity`, `B/I-cuisine`, `B/I-dish`, `B/I-hours`, `B/I-location`, `B/I-price`, `B/I-rating`, and `B/I-restaurant-name`.

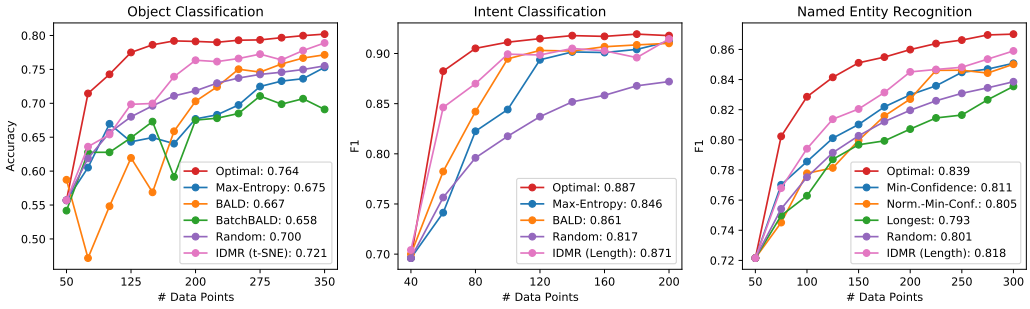

Figure 9: IDMR-augmented heuristics performance, improving upon the vanilla versions by 2.95% on average.

tences. This is indeed exploited by the min-confidence heuristic, where very long sentences are vastly over-sampled. However, the optimal order does not exhibit such length-seeking behavior, and instead matches the test set distribution quite closely.

6.5 Distribution-Matching Regularization

Sec. 6.4 consistently suggests that unlike heuristics, the optimal order matches the data distribution meticulously. Can we improve heuristics with this insight?

Algorithm 2: Input Dist.-Matching Reg. (IDMR)

Input: $\mathcal{A}\left(m_\theta, \mathcal{D}^L, \mathcal{D}^U\right)$ that returns the next data point in $\mathcal{D}^U$ to label
$d_{\text{ref}} = \texttt{bin-distribution}\left(\mathcal{D}^L_{0,X} \cup \mathcal{D}^U_X \cup \mathcal{D}^M\right)$;
$d_{\text{cur}} = \texttt{bin-distribution}\left(\mathcal{D}^L_X\right)$;
$b^* = \arg\min_b \left(d_{\text{cur}} - d_{\text{ref}}\right)_b$;
$\mathcal{D}^U_{b^*} = \{x \in \mathcal{D}^U_X : \texttt{bin}(x) = b^*\}$;
return $\mathcal{A}\left(m_\theta, \mathcal{D}^L, \mathcal{D}^U_{b^*}\right)$

Alg. 2 presents Input Distribution-Matching Regularization (IDMR), a meta-algorithm that augments any acquisition function to its input distribution-matching regularized version. $\texttt{bin}(x)$ is a function that assigns each input data point x to one of finitely many bins by its characteristics. $\texttt{bin-distribution}(\mathcal{D}_X)$ computes the empirical count distribution of the bin values for input part of $\mathcal{D}_X$. The IDMR algorithm compares the reference bin distribution on all available data (i.e. $\mathcal{D}^L_{0,X} \cup \mathcal{D}^U_X \cup \mathcal{D}^M_X$) to the current bin distribution of $\mathcal{D}^L_X$, and runs the base acquisition function on the fraction of data points in the pool set with the value of the highest deficit bin.

For object classification, a K-means algorithm was used to find five clusters in the t-SNE space, used as the five bins. For intent classification and NER, we binned the data according to sentence length. We used IDMR to augment the best performing heuristics. As shown in Fig. 9, on all tasks, the IDMR-augmented heuristic is better than all but the optimal order, with an average improvement of 2.49% over the its non-augmented counterpart.

We also tried using a similar technique to match the output label distribution but observed mixed results. The algorithm and results are described in App. H.

7 Discussion and Conclusion

In this paper, we searched for and analyzed the optimal data acquisition order in active learning. Our findings are consistent across tasks and models. First, we confirmed the dependence of optimal orders on training stochasticity such as model initialization or dropout masking, which should be explicitly considered as AL methods approach the optimal limit.

Notably, we did not find any evidence that the optimal order needs to over/under-sample in hard/easy-to-learn regions. Given this, it is reasonable that the optimal order transfers across architectures better than heuristics and that heuristics can benefit from a distribution-matching regularization.

Indeed, most AL heuristics seek to optimize proxy objectives such as maximal informativeness or problem space coverage. While intuitive, they have not been rigorously justified to correlate with AL performance (e.g. ξ-quality). For example, even if a data point is maximally informative, it is possible that the information would not be fully absorbed during training and reflected in performance improvement.

Moreover, in supervised learning, the fundamental assumption that the training and test data come from the same underlying distribution is the key to most guarantees of empirical risk minimization (ERM). Thus, we conjecture that the distribution-matching property arises from the nice behavior of ERM under this assumption. This also suggests that when the proposed algorithm is expected to violate this assumption, more careful analysis should be done on how such distribution shift may (adversely) affect the performance of models optimized under ERM. Finally, developing theoretical guarantees of ERM under controlled distribution mismatch and/or formulations beyond ERM may also benefit active learning.

References

Bachman, P., Sordoni, A., and Trischler, A. (2017). Learning algorithms for active learning. *arXiv preprint arXiv:1708.00088*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Chen, X., Ghoshal, A., Mehdad, Y., Zettlemoyer, L., and Gupta, S. (2020). Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*.

Contardo, G., Denoyer, L., and Artières, T. (2017). A meta-learning approach to one-step active learning. *arXiv preprint arXiv:1706.08334*.

Fang, M., Li, Y., and Cohn, T. (2017). Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. (2007). Active learning with gaussian processes for object categorization. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

Kasai, J., Qian, K., Gurajada, S., Li, Y., and Popa, L. (2019). Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirsch, A., van Amersfoort, J., and Gal, Y. (2019). Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pages 7026–7037.

Konyushkova, K., Sznitman, R., and Fua, P. (2017). Learning active learning from data. In *Advances in Neural Information Processing Systems*, pages 4225–4235.

Konyushkova, K., Sznitman, R., and Fua, P. (2018). Discovering general-purpose active learning strategies. *arXiv preprint arXiv:1810.04114*.

Koshorek, O., Stanovsky, G., Zhou, Y., Srikumar, V., and Berant, J. (2019). On the limits of learning to actively learn semantic representations. *arXiv preprint arXiv:1910.02228*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

Liu, J., Pasupat, P., Cyphers, S., and Glass, J. (2013). ASGARD: A portable architecture for multilingual dialogue systems. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390. IEEE.

Liu, M., Buntine, W., and Haffari, G. (2018). Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lowell, D., Lipton, Z. C., and Wallace, B. C. (2018). Practical obstacles to deploying active learning. *arXiv preprint arXiv:1807.04801*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Pang, K., Dong, M., Wu, Y., and Hospedales, T. (2018). Meta-learning transferable active learning policies by deep reinforcement learning. *arXiv preprint arXiv:1806.04798*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Roy, N. and McCallum, A. (2001). Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448.

Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Shen, Y., Yun, H., Lipton, Z. C., Kronrod, Y., and Anandkumar, A. (2017). Deep active learn-

ing for named entity recognition. *arXiv preprint arXiv:1707.05928*.

Sinha, S., Ebrahimi, S., and Darrell, T. (2019). Variational adversarial active learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5972–5981.

Vu, T., Liu, M., Phung, D., and Haffari, G. (2019). Learning how to active learn by dreaming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4091–4101.

Woodward, M. and Finn, C. (2017). Active one-shot learning. *arXiv preprint arXiv:1702.06559*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xu, Z., Akella, R., and Zhang, Y. (2007). Incorporating diversity and density in active learning for relevance feedback. In *European Conference on Information Retrieval*, pages 246–257. Springer.

A Meta-Analysis of Existing Results

In this section, we present a meta-analysis of existing results to highlight the inconsistencies among them. Fig. 1 of (Gal et al., 2017) (replicated as Fig. 10 left) shows that BALD (Bayesian active learning by disagreement) using MC-dropout achieves much higher performance than the random baseline on one image dataset. By contrast, Fig. 2 of (Sinha et al., 2019) (replicated as Fig .10 right) shows that MC-Dropout methods struggle to outperform even the random baseline on four datasets.

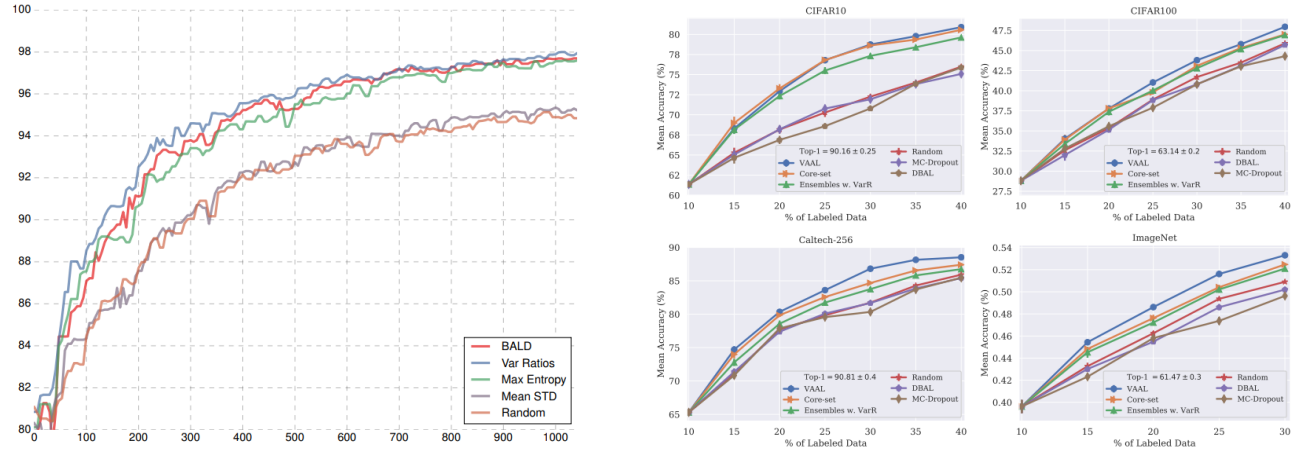

Figure 10: Left: Fig. 1 of (Gal et al., 2017). Right: Fig. 2 of (Sinha et al., 2019).

Fig. 2 of (Gal et al., 2017) (replicated as Fig. 11 top) shows that active learning strategies with MC-Dropout-based uncertainty estimation consistently outperforms their deterministic counterparts on a CV task. However, Fig. 4 of (Shen et al., 2017) (replicated as Fig. 11 bottom) finds no discernible difference between MC-Dropout-based BALD and other deterministic heuristics on an NLP task.

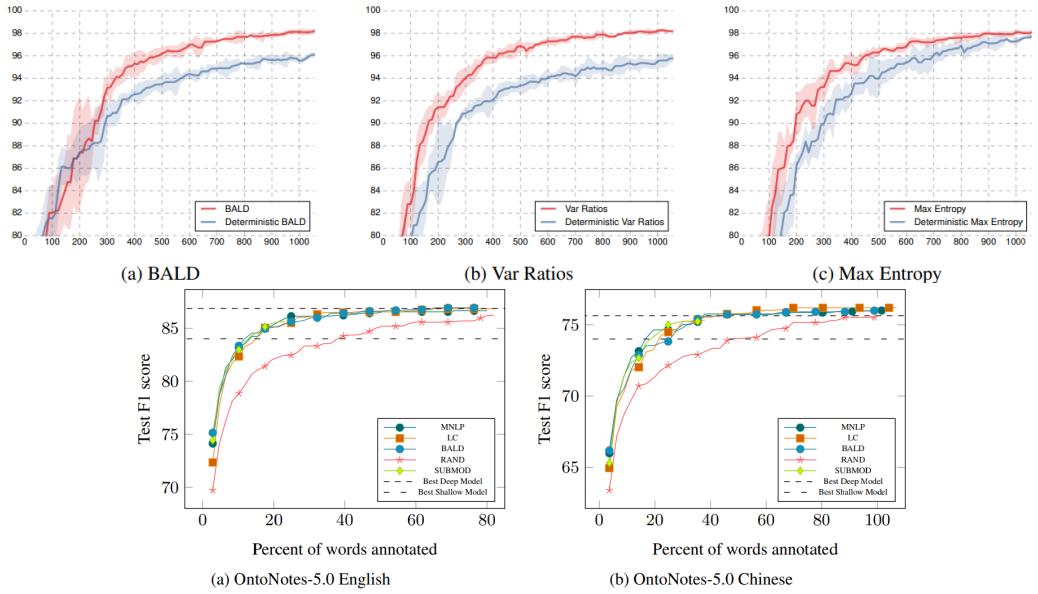

Figure 11: Top: Fig. 2 of (Gal et al., 2017). Bottom: Fig. 4 of (Shen et al., 2017).

For meta-active-learning methods, Fig. 3 of (Fang et al., 2017) (replicated as Fig. 12 top) shows that the RL-based PAL (policy active learning) consistently outperforms uncertainty and random baselines in a cross-lingual transfer setting. However, Fig .2 of (Liu et al., 2018) (replicated as Fig. 12 middle) shows that PAL are hardly better than uncertainty or random baselines in both cross-domain and cross-lingual transfer settings, while their proposed ALIL (active learning by imitation learning) is better than all studied baselines. Nevertheless, Fig. 4

of (Vu et al., 2019) (replicated as Fig. 12 bottom) again shows that both PAL and ALIL are not better than the uncertainty-based heuristic on a named entity recognition task, while their proposed active learning by dreaming method ranks the best.

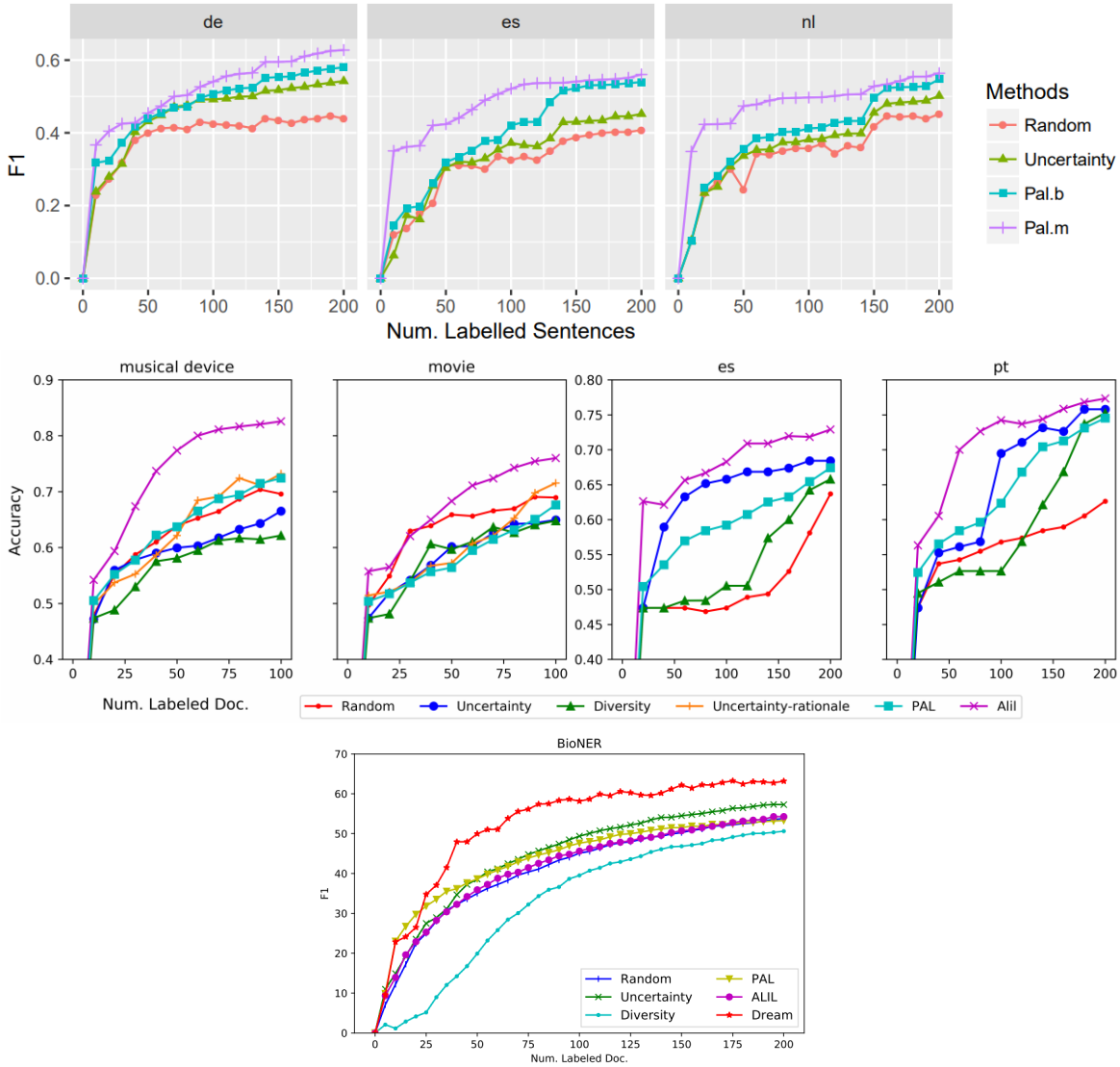

Figure 12: Top: Fig. 3 of (Fang et al., 2017). Middle: Fig .2 of (Liu et al., 2018). Bottom: Fig. 4 of (Vu et al., 2019).

These results demonstrate concerning inconsistency and lack of generalization to even very similar datasets or tasks. In addition, the above-presented performance curves are typically the sole analysis conducted by the authors. Thus, we believe that it is enlightening to analyze more aspects of a proposed strategy and its performance. In addition, a comparison with the optimal strategy could reveal actionable ways to improve the proposed strategy, which is the main motivation of our work.

B Considerations for Stochastic Acquisition Functions

It is straightforward to extend the discussion in Sec. 3 to stochastic acquisition functions, such as those that require stochastic uncertainty estimation, like BALD, or those that are represented as stochastic policies.

First, we introduce ζ to capture such randomness, so that $\Delta\mathcal{D}^L_{k+1} = \mathcal{A}(\theta_k, \mathcal{D}^L_k, \zeta)$ remains deterministic. Then we can establish a one-to-one relationship between $\mathcal{A}(\cdot, \cdot, \zeta)$ and order σ_ζ on $\mathcal{D}^U$. By definition of ξ-optimal order σ^*_ξ, we still have $q_\xi(\sigma^*_\xi) \geq q_\xi(\sigma_\zeta)$, and $q_\xi(\sigma^*_\xi) \geq \mathbb{E}_\zeta\left[q_\xi(\sigma_\zeta)\right]$. In other words, the ξ-optimal order σ^*_ξ outperforms

both the expected quality of the stochastic acquisition function and any single instantiation of stochasticity.

The above discussion suggests that not accounting or for stochasticity in acquisition function or unable to do so (i.e. using the expected quality) is inherently sub-optimal. Since many heuristics require uncertainty estimation through explicitly stochastic processes such as MC-Dropout (Gal and Ghahramani, 2016), the stochastic components may become the bottleneck to further improvement as AL algorithms approach the optimal limit.

C Model Architecture

Input: $28 \times 28 \times 1$
Conv: 32 5×5 filters
(Optional: MC-Dropout)
Max-Pool: 2×2
ReLU
Conv: 64 5×5 filters
(Optional: MC-Dropout)
Max-Pool: 2×2
ReLU
Linear: 1024×128
(Optional: MC-Dropout)
ReLU
Linear: 128×10

(a) CNN Object Class.

Input: L tokens
GloVe embedding: 300 dim
BiLSTM: 300 hidden dim
Average of L hiddens
(Optional: MC-Dropout)
Linear: 600×7

(b) LSTM Intent Class.

Input: L tokens
GloVe embedding: 300 dim
Conv: 50 size-3 kernels
ReLU
Conv: 50 size-3 kernels
ReLU
Conv: 50 size-3 kernels
ReLU
Average-Pool
Linear: 50×50
ReLU
(Optional: MC-Dropout)
Linear: 50×7

(c) CNN Intent Class.

Input: L tokens
GloVe embedding: 300 dim
Averege of L embeddings
Linear: 300×100
ReLU
Linear: 100×100
ReLU
(Optional: MC-Dropout)
Linear: 100×7

(d) AOE Intent Class.

Table 3: Model Architectures for Object and Intent Classification

C.1 Object Classification

We follow the architecture used by Kirsch et al. (2019). The MC-Dropout layers are present in BALD and BatchBALD for uncertainty estimation. The architecture is summarized in Tab. 3(a).

C.2 Intent Classification

We use an LSTM architecture as our main architecture of study for the intent classification task. We initialize the word embeddings with GloVe (Pennington et al., 2014) and jointly train it along with other model parameters. The architecture is shown in Tab. 3(b). Again, MC-Dropout is only used for uncertainty estimation in BALD.

To study model transfer performance, we also used (1D-)CNN and Average of Embedding (AOE) architectures, shown in Tab. 3(c) and (d), as well as the pre-trained RoBERTa architecture (Liu et al., 2019) with a linear classification layer on top.

C.3 Named Entity Recognition

For named entity recognition, we used the same architecture as Shen et al. (2017), except that we removed character-level embedding layer since an ablation study did not find it beneficial to the performance.

Specifically, the model architecture is encoder-decoder. The encoder builds the representation for each word in the same way as the BiLSTM architecture for intent classification. At each decoding step, the decoder maintains the running hidden and cell state, receives the concatenation of the current word encoder representation and previous tag one-hot representation as the input, and predicts the current tag with a linear layer from output hidden state. At the first step, a special `[GO]` tag was used.

D Extended Quality Summary

Tab. 4 presents the numerical quality scores for optimal, heuristic, and random orders. The object and intent classification tasks are evaluated and averaged across 5 different model seeds, and the NER task is evaluated on

a single model seed. In all cases, we can observe a large gap between the optimal order and the rest, indicating abundant room of improvement for the heuristics.

	Object Class.	Intent Class.		NER
Optimal	0.761	0.887	Optimal	0.838
Max-Entorpy	0.682	0.854	Min-Confidence	0.811
BALD	0.676	0.860	Norm. Min-Conf.	0.805
BatchBALD	0.650	N/A	Longest	0.793
Random	0.698	0.816	Random	0.801

Table 4: Complete results of optimal, heuristic, and random order qualities.

E Optimal Performance Curves

Fig. 13 shows the performance curves on the validation and test sets for intent classification (left two panels) and named entity recognition (right two panels). Similar to those for object classification (Fig. 3), the shape of the curves are quite similar on validation and test sets, indicating good generalization.

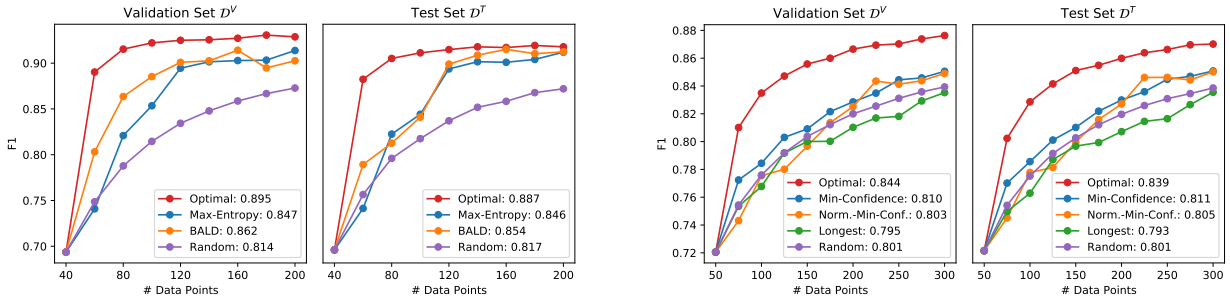

Figure 13: Performance curves for intent classification (left two panels) and NER (right two panels).

F Relative Order Ranking Between Architectures

Fig. 14 shows how the optimal (top) and heuristic (bottom) orders for different architectures relate to each other on the intent classification task. Specifically, each vertical bar is composed of 160 horizontal stripes, which represent the 160 data points acquired under the specified order, out of a pool of 800 data points (not shown). The first stripe from the top represents the first data point being acquired, followed by data points represented by the second, third, etc. stripes from the top. A line between the two orders connects the same data point in both orders and reflects the position of that data point in each of the two orders. The title shows the number of shared data points that are acquired by both orders (i.e. number of connecting lines).

Paradoxically, we find less sharing between optimal orders than between heuristic orders, even though the transfer learning performance is generally better for the optimal orders. This suggests that there are an abundant amount of high quality data points that could lead to high AL performance, despite the optimal order for different architectures select largely different ones. However, they are mostly undiscovered by the heuristics, which share heavily between each other because the over-sampling of data points that are short in length or belong to the minority class depletes the pool of such examples.

G Additional Visualizations for Object Classification

Fig. 15 presents the input (via PCA and t-SNE dimensionality reduction) and output distribution for BALD and BatchBALD order. The labels for data points sampled by the BALD and BatchBALD heuristics are much less balanced than the optimal and random orders. In addition, BatchBALD seems to focus on only a few concentrated regions in the input space.

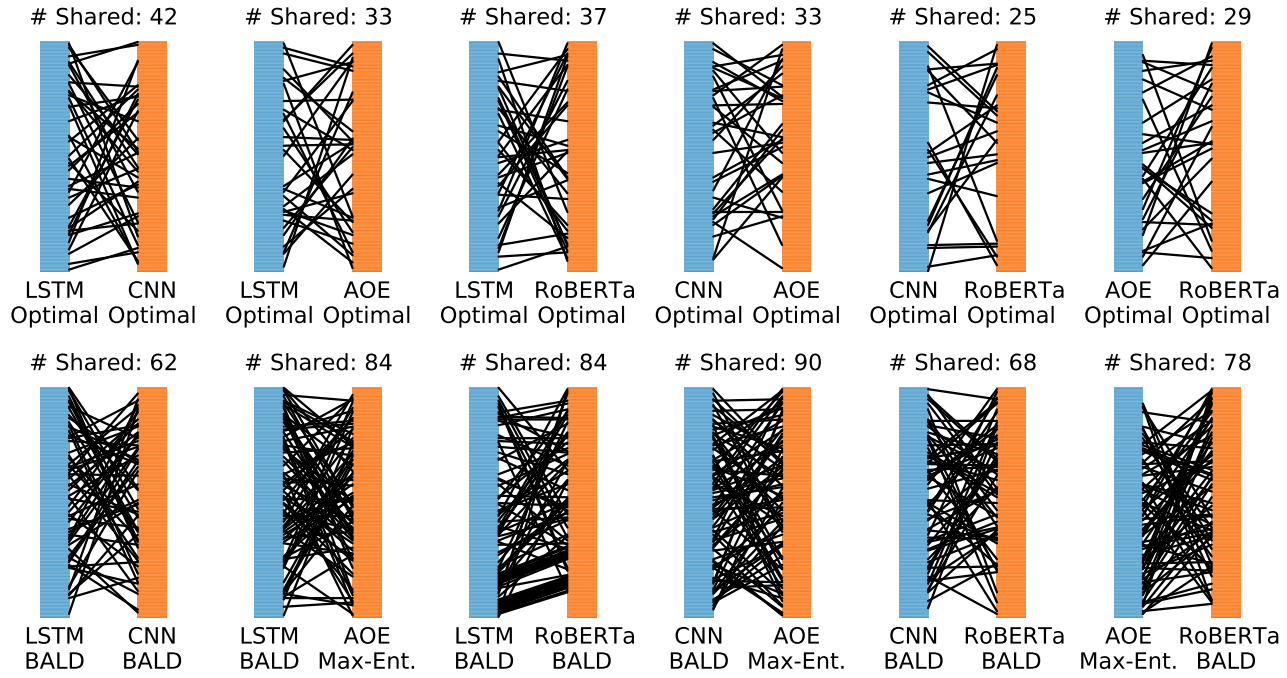

Figure 14: A visual comparison of optimal (top) and heuristic (bottom) orders, for every pair of architectures, showing the number of shared data points acquired under both architectures and their relative ranking.

H Output Distribution-Matching Regularization (ODMR)

In this section we describe the mixed results of regularizing the output distribution on object and intent classification. Since NER is a structured prediction task where the output tags in a sentence are correlated (e.g. I-tags always follow B- or I-tags), it is not clear what is the best way to enforce the distribution-matching property in the tag space and we leave the investigation to future work.

Alg. 3 presents the OMDR algorithm. Compared to the IDMR algorithm (Alg. 2), there are two additional challenges. First, the labels for the pool set are not observed, so when the algorithm tries to acquire a data point with a certain label, it can only use the predicted label, which, with few data, can be very wrong in some cases. As soon as a data point is selected for annotation, its label is revealed immediately and used to calculate d_{cur} during the next data point selection. In other words, data annotations are not batched.

Second, due to the unavailability of pool labels, the only labels for which we can use to compute the reference distribution are from $\mathcal{D}_0^L$ and $\mathcal{D}^M$. If they are small, as are typically, the reference label distribution can be far from the true distribution $\mathbb{P}_Y$, especially for rare classes. Thus, we employ the add-1 smoothing (i.e. adding 1 to each label count before calculating the empirical count distribution, also known as Laplace smoothing) to ensure that rare classes are not outright missed.

Algorithm 3: Output Distribution-Matching Regularization (IDMR)

Input: $\mathcal{A}\left(\mathcal{D}^U, m_\theta, \mathcal{D}^L\right)$ that returns the next data point in $\mathcal{D}^U$ to label
$d_{\text{ref}} = \texttt{label-distribution}\left(\mathcal{D}^L_{0,Y} \cup \mathcal{D}^M_Y, \texttt{add-one=True}\right)$;
$d_{\text{cur}} = \texttt{label-distribution}\left(\mathcal{D}^L_Y, \texttt{add-one=False}\right)$;
$l^* = \arg\min_l \left(d_{\text{cur}} - d_{\text{ref}}\right)_l$;
$\mathcal{D}^U_{l^*} = \{x \in \mathcal{D}^U : m_\theta(x) = l^*\}$;
return $\mathcal{A}\left(\mathcal{D}^U_{b^*}, m_\theta, \mathcal{D}^L\right)$

In addition to Alg. 3, we also experiment with three "cheating" versions of ODMR. The most extreme version (test+groundtruth) uses test set (rather than the accessible set $\mathcal{D}_0^L \cup \mathcal{D}^M$) to estimate label distribution and the groundtruth label (rather than the predicted label) are used to construct the subset $\mathcal{D}^U_{l^*}$. The other two versions

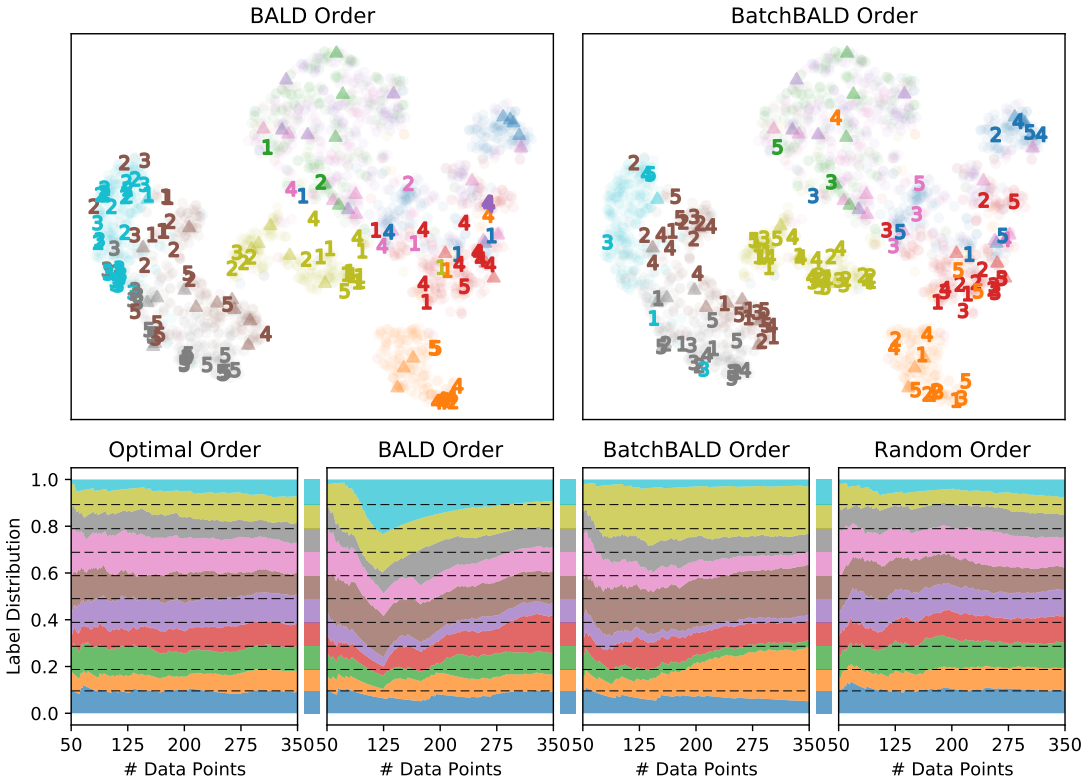

Figure 15: Top: the first five batches numerically labeled in t-SNE embedding space, along with warmstart (triangle) and test (circle) samples. Bottom: label distribution w.r.t. labeled set size, with test-set distribution shown between plots and as dashed lines.

(accessible+groundtruth and test+predicted) drops either piece of information. The actual "non-cheating" ODMR (accessible+predicted) is the only version that is practically feasible in reality.

Fig. 16 presents the four versions of ODMR-augmented heuristics on the intent classification result. We can see that all four versions are able to outperform the baseline heuristic, and three out of four (including the "non-cheating" ODMR) are able to outperform the random baseline. In addition, the label distribution are much more uniform compared to the unregularized version in Fig. 6 (bottom middle).

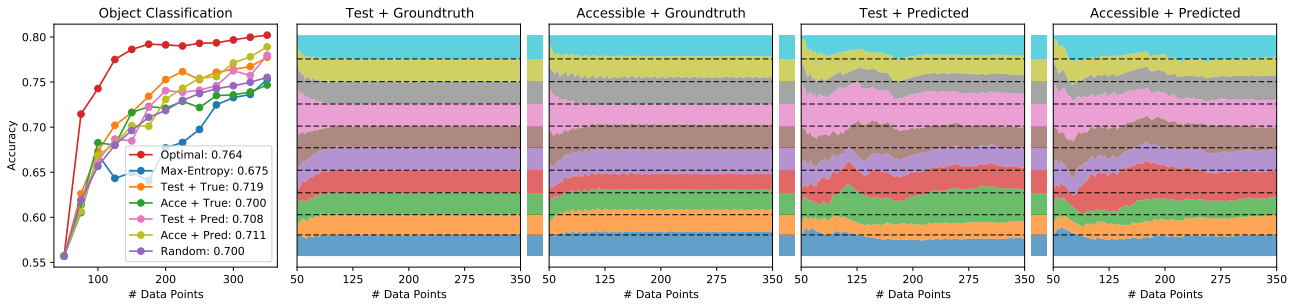

Figure 16: ODMR with max-entropy heuristic on object classification along with observed label distribution.

Fig. 17 presents the four versions of ODMR-augmented max-entropy (top) and BALD (bottom) heuristics on the intent classification result. Unlike object classification, none of the "cheating" or "non-cheating" versions are able to outperform the respective vanilla heuristic. Interestingly, using predicted rather than groundtruth labels achieves better performance in both cases.

Overall, the results are inconclusive about whether ODMR can help. Since it uses the predicted label to guide

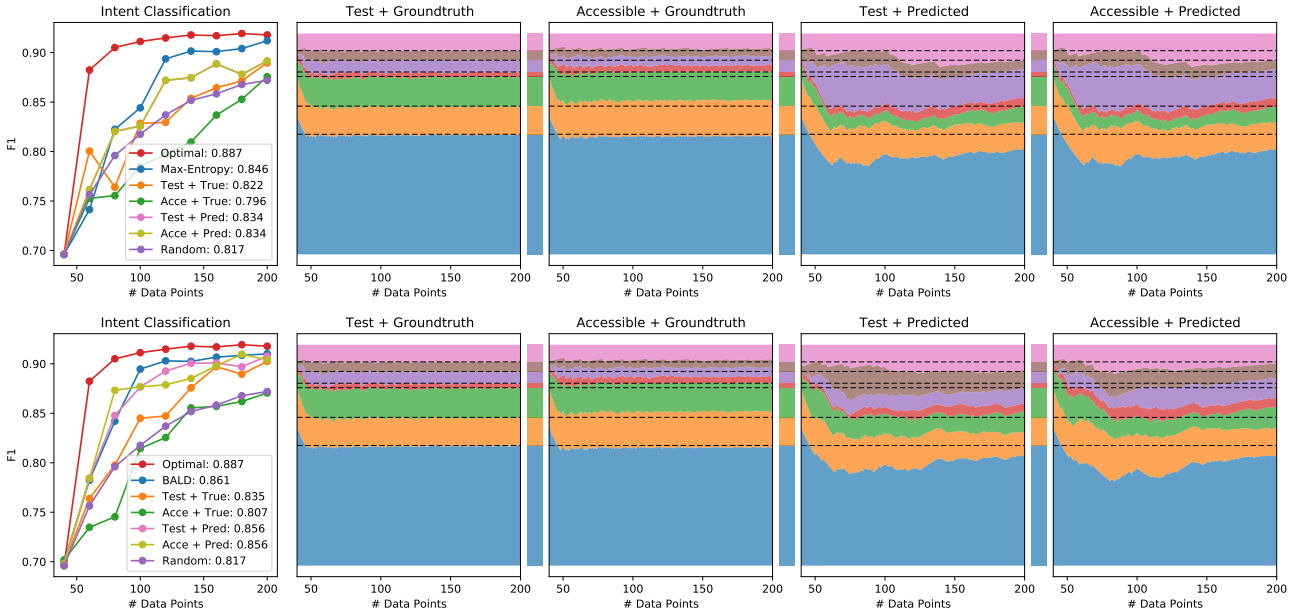

Figure 17: ODMR with max-entropy (top) and BALD (bottom) heuristics on intent classification along with observed label distribution.

data acquisition, this kind of bootstrapping is prone to "confident mistakes" when a data point is predicted wrong but with high certainty and thus never acquired, while the selection mechanism of IDMR provides a much more independent, yet still heuristic-guided, way to cover the data distribution. Visually, ODMR performance curves for both tasks are much less smooth than IDMR curves (Fig. 9), indicating unstability of using output labels to regularize a prediction task. Additional studies are required to determine when and how ODMR can help, and we leave them to future work.