Deep Bayesian Active Learning via Equivalence Class Annealing

Renyu Zhang ZHANGR@UCHICAGO.EDU
University of Chicago

Aly A. Khan AAKHAN@UCHICAGO.EDU
University of Chicago

Robert L. Grossman RGROSSMAN1@UCHICAGO.EDU
University of Chicago

Yuxin Chen CHENYUXIN@UCHICAGO.EDU
University of Chicago

Abstract

Active learning has demonstrated data efficiency in many fields. Existing active learning algorithms, especially in the context of deep Bayesian active models, rely heavily on the quality of uncertainty estimations of the model. However, such uncertainty estimates could be heavily biased, especially with limited and imbalanced training data. In this paper, we propose BALanCe, a deep Bayesian active learning framework that mitigates the effect of such biases. Concretely, BALanCe employs a novel acquisition function which leverages the structure captured by equivalence hypothesis classes and facilitates differentiation among different equivalence classes. Intuitively, each equivalence class consists of instantiations of deep models with similar predictions, and BALanCe adaptively adjusts the size of the equivalence classes as learning progresses. Besides the fully sequential setting, we further propose Batch-BALanCe—a generalization of the sequential algorithm to the batched setting—to efficiently select batches of training examples that are jointly effective for model improvement. We show that Batch-BALanCe achieves state-of-the-art performance on several benchmark datasets for active learning, and that both algorithms can effectively handle realistic challenges that often involve multi-class and imbalanced data.

Keywords: Deep Active Learning, Bayesian Neural Network, Importance Sampling

1. Introduction

Active learning (AL) (Cohn et al., 1996; Tong and Koller, 2001; Dasgupta and Langford, 2011; Settles, 2012) characterizes a collection of techniques that efficiently select data for training machine learning models. In a prototypical *pool-based* setup, an active learner selectively queries the labels of data points from a pool of unlabeled examples, and incurs a certain cost for each label obtained. The goal is to minimize the total cost while achieving a target level of performance. A common practice for active learning is to devise efficient surrogates, aka *acquisition functions*, to assess the effectiveness of unlabeled data points in the pool. There have been a vast body of literature and empirical studies (Huang et al., 2010; Houlsby et al., 2011; Wang and Ye, 2015; Hsu and Lin, 2015; Huang et al., 2016; Sener and Savarese, 2017; Ducoffe and Precioso, 2018; Ash et al., 2019; Liu et al., 2020;

Yan et al., 2020) suggesting a variety of heuristics as potential acquisition functions for AL. Among these methods, *Bayesian Active Learning by Disagreement* (BALD) (Houlsby et al., 2011) has attained notable success in the context of deep Bayesian AL, while maintaining the expressiveness of Bayesian models (Gal et al., 2017; Janz et al., 2017; Shen et al., 2017).

BALD greedily queries the data point that has the maximal *mutual information* with the model parameters at each iteration. As discussed in §A.1, we show that there are pessimistic scenarios where BALD may fail. These examples are generally applicable to information-theoretic query strategies (Golovin et al., 2010). Inspired by Chen et al. (2016), we propose a novel deep active learning framework with Bayesian neural networks (BNNs) based on a decision-theoretic criterion. Our key contributions are highlighted below.

1. We propose BALanCe, a novel deep Bayesian AL framework that leverages disagreement structures captured by *equivalence classes* over candidate models (see §2 for its formal definition). Intuitively, an equivalence class consists of models that (approximately) agree on their predictions over *i.i.d.* unlabeled samples from the target distribution.[1] By annealing the diameter of equivalence classes BALanCe can adapt to task difficulty at different stages of AL (§3.1).
2. We extend BALanCe to the batched setting, and propose efficient approximations to handle the selections of queries in batches. The resulting algorithm, namely Batch-BALanCe, which is based on a novel importance sampling strategy, can efficiently scale to realistic batched learning tasks with reasonably large batch sizes (§3.2,§C).
3. We demonstrate the effectiveness of the proposed algorithms via an extensive empirical study. Batch-BALanCe achieves state-of-the-art performance—sometimes by a large margin—on several benchmark datasets involving challenging distributions such as multi-class and imbalanced data (§4, §F).

2. Background and problem setup

Notations We consider pool-based Bayesian AL, where we are given an unlabelled dataset $\mathcal{D}_{\text{pool}}$ drawn *i.i.d.* from some underlying data distribution. Further, assume a labeled dataset $\mathcal{D}_{\text{train}}$ and a set of hypotheses $\mathcal{H} = \{h_1, \ldots, h_n\}$. We would like to distinguish a set of (unknown) target hypotheses among the ground set of hypotheses $\mathcal{H}$. Let H denote the random variable that represents the target hypotheses. Let $\mathrm{p}(H)$ be a prior distribution over the hypotheses. In this paper, we resort to BNN with parameters $\omega \sim \mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})$[2].

Problem statement An AL algorithm will select samples from $\mathcal{D}_{\text{pool}}$ and query labels from experts. The experts will provide label y for given query $x \in \mathcal{D}_{\text{pool}}$. We assume labeling each query x incurs a unit cost. Our goal is to find an adaptive policy for selecting samples that allows us to find a hypotheses with target error rate $\sigma \in [0, 1]$ while minimizing the total cost of the queries. Formally, a *policy* π is a mapping π from the labeled dataset $\mathcal{D}_{\text{train}}$ to samples in $\mathcal{D}_{\text{pool}}$. We use $\mathcal{D}^{\pi}_{\text{train}}$ to denote the set of examples chosen by π. Given the labeled dataset $\mathcal{D}^{\pi}_{\text{train}}$, we define $\mathrm{p}_{\text{ERR}}(\pi)$ as the expected error probability w.r.t. the

1. Our selection criterion takes into account the informativeness of a query with respect to the target models, while putting less focus on differentiating models with little disagreement on target data distribution.
2. We use the conventional notation ω to represent the parameters of a BNN, and use ω and h interchangeably to denote a hypothesis.

posterior $\mathrm{p}(\omega \mid \mathcal{D}^{\pi}_{\text{train}})$. Let the cost of a policy π be $\text{cost}(\pi) \triangleq \max |\mathcal{D}^{\pi}_{\text{train}}|$, i.e., the maximum number of queries made by policy π over all possible realizations of the target hypothesis $H \in \mathcal{H}$. Given a tolerance parameter $\sigma \in [0, 1]$, we seek a policy with the minimal cost, such that upon termination, it will get expected error probability less than σ. Formally, we seek $\arg\min_{\pi} \text{cost}(\pi)$, s.t. $\mathrm{p}_{\text{ERR}}(\pi) \leq \sigma$.

The equivalence-class-based selection criterion A strategy in approximate AL involves learning *equivalence classes* (ECs) (Golovin et al., 2010). In the following, we review the concept and introduce the problem setup for EC-based selection criteria. We defer the detailed discussion of the limitations of EC-based AL algorithms to §B.1 and §B.2.

Definition 1 (Equivalence Class) *Let $(\mathcal{H}, d)$ be a metric space where $\mathcal{H}$ is a hypothesis class and d is a metric. For a given set $\mathcal{V} \subseteq \mathcal{H}$ and centers $\mathcal{S} = \{s_1, ..., s_k\} \subseteq \mathcal{V}$ of size k, let $r^{\mathcal{S}} : \mathcal{V} \to [k]$ be a partition function over $\mathcal{V}$ and $\mathcal{D}_i := \{h \in \mathcal{V} \mid r^{\mathcal{S}}(h) = i\}$, such that $\forall i, j \in [k], r^{\mathcal{S}}(s_i) = i$ and $\forall h \in \mathcal{D}_i, d(h, s_i) \leq d(h, s_j)$. Each $\mathcal{D}_i \subseteq \mathcal{V}$ is called an equivalence class induced by $s_i \in \mathcal{S}$.*

Consider a pool-based AL problem with hypothesis space $\mathcal{H}$, a sampled set $\mathcal{V} \subseteq \mathcal{H}$, and an unlabeled dataset $\bar{\mathcal{D}}_{\text{pool}}$ which is drawn i.i.d. from the underlying data distribution. Each hypothesis $h \in \mathcal{H}$ can be represented by a vector v_h indicating the predictions of all samples in $\bar{\mathcal{D}}_{\text{pool}}$. We can construct equivalence classes with the Hamming distances, which are denoted as $d_{\text{H}}(h, h')$ and are calculated based on the predictions of $\bar{\mathcal{D}}_{\text{pool}}$, and equivalence class number k on sampled hypotheses $\mathcal{V}$. Let $d^{\mathcal{S}}_{\text{H}}(\mathcal{V}) := \max_{h, h' \in \mathcal{V} : r^{\mathcal{S}}(h) = r^{\mathcal{S}}(h')} d_{\text{H}}(h, h')$ be the maximal diameter of equivalence classes induced by $\mathcal{S}$. Therefore, the error rates of any unordered pair of hypotheses $\{h, h'\}$ that lie in the same equivalence class are at most $d^{\mathcal{S}}_{\text{H}}(\mathcal{V})$ away from each other. If we construct the k equivalence-class-inducing centers (as in Definition 1) as the solution of the max-diameter clustering problem: $\mathcal{C} = \arg\min_{|\mathcal{S}|=k} d^{\mathcal{S}}_{\text{H}}(\mathcal{V})$, we can obtain the minimal worst-case relative error (i.e., difference in error rate) between hypotheses pair $\{h, h'\}$ that lie in the same equivalence class. We denote $\mathcal{E} = \{\{h, h'\} : r^{\mathcal{C}}(h) \neq r^{\mathcal{C}}(h')\}$ as the set all (unordered) pairs of hypotheses (i.e., undirected edges) corresponding to different equivalence classes when the centers are $\mathcal{C}$.

3. Our Approach

3.1 The BALanCe algorithm

In this paper, we resort to BNNs and MC dropout (Gal and Ghahramani, 2016) to estimate the ECED objective Δ_{ECED}. We train the BNN with all available labeled samples $\mathcal{D}_{\text{train}}$ at each iteration. Hypotheses ω are sampled from the BNN posterior by MC dropout (Gal and Ghahramani, 2016). In order to determine if there is an edge $\{\hat{\omega}, \hat{\omega}'\}$ that connects a pair of sampled hypotheses $\hat{\omega}, \hat{\omega}'$ (i.e., if they are in different equivalence classes), we calculate the Hamming distance $d_{\text{H}}(\hat{\omega}, \hat{\omega}')$ between the predictions of $\hat{\omega}, \hat{\omega}'$ on the unlabeled dataset $\bar{\mathcal{D}}_{\text{pool}}$. If the distance is greater than some threshold τ, we consider the edge $\{\hat{\omega}, \hat{\omega}'\} \in \hat{\mathcal{E}}$; otherwise not. With BNNs, we can safely remove the offset term $\left(1 - \max_{\omega} \lambda^2_{\omega,y}\right)$ since for a reasonable sized model, $\max_{\omega} \lambda^2_{\omega,y}$ is close or equal to 1. We define the acquisition function of BALanCe as:

$$\Delta_{\text{BALanCe}}(x \mid \mathcal{D}_{\text{train}}) \triangleq \mathbb{E}_y \mathbb{E}_{\omega, \omega' \sim \mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{1}_{d_{\text{H}}(\omega, \omega') > \tau} \cdot \left(1 - \lambda_{\omega,y} \lambda_{\omega',y}\right) \tag{1}$$

where $\mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k)>\tau}$ is the indicator function which equals to 1 only when $d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k) > \tau$. In practice, the threshold τ is not known. Instead of using a fixed parameter τ, we adaptively anneal τ by setting τ proportional to BNN's validation error rate in each AL iteration. BALanCe iterates until the trained BNN achieves the target accuracy.

In practice, we cannot directly compute the expectation Eq. (1); instead we run Alg. 1 as an approximation: We first acquire K pairs of MC dropout samples $\{\hat{\omega},\hat{\omega}'\}$. The Hamming distances $d_{\mathrm{H}}(\hat{\omega},\hat{\omega}')$ between these pairs of BNN MC dropout samples are computed. Next, we calculate the weight discount factor $1-\lambda_{\hat{\omega}_k,y}\lambda_{\hat{\omega}'_k,y}$ for each possible label y and each pair $\{\hat{\omega},\hat{\omega}'\}$ where $d_{\mathrm{H}}(\hat{\omega},\hat{\omega}') > \tau$. At last, we take the expectation of the discounted weight over all y configurations. In summary, $\Delta_{\mathrm{BALanCe}}$ is approximated as follows:

$$\Delta_{\mathrm{BALanCe}}(x \mid \mathcal{D}_{\mathrm{train}}) \approx \sum_{\hat{y}}\sum_{k=1}^{K}\frac{\mathrm{p}(\hat{y}\mid\hat{\omega}_k)+\mathrm{p}(\hat{y}\mid\hat{\omega}'_k)}{2K}\sum_{k=1}^{K}\frac{\mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k)>\tau}\cdot\left(1-\lambda_{\hat{\omega}_k,\hat{y}}\lambda_{\hat{\omega}'_k,\hat{y}}\right)}{K}.$$

Note that in our algorithms we never explicitly construct equivalence classes on MC dropout samples, due to the fact that (1) it is intractable to find the exact solution for the max-diameter clustering problem (see §2) and (2) an explicit approximate partitioning of the hypotheses samples tends to introduce "unnecessary" edges where the incident hypotheses are closeby (e.g., if a pair of hypotheses lie on the adjacent edge between two hypothesis partitions), and therefore may overly estimate the utility of a query. Nevertheless, we conducted an empirical study of a variant of BALanCe with explicit partitioning (which underperforms BALanCe). We defer detailed discussion on this approach, as well as empirical study, to §F.3.

3.2 Batch-BALanCe: Batch-mode AL

We now consider a batch-mode variant of BALanCe, via a novel acquisition function $\Delta_{\mathrm{Batch-BALanCe}}$ by generalizing $\Delta_{\mathrm{BALanCe}}$ to subsets of points $x_{1:B}$, where $x_{1:B} \triangleq \{x_1,\ldots,x_B\}$ and B is the batch size. To avoid the combinatorial explosion when evaluating subsets, we compute the acquisition function $\Delta_{\mathrm{Batch-BALanCe}}(\{x_1,\ldots,x_B\} \mid \mathcal{D}_{\mathrm{train}})$ in a greedy manner. In each acquisition iteration b inside the batch, the acquisition function becomes

$$\begin{aligned}&\Delta_{\mathrm{Batch-BALanCe}}(x_{1:b} \mid \mathcal{D}_{\mathrm{train}})\\ &\approx \sum_{\hat{y}_{1:b}}\left(\sum_{k=1}^{K}\frac{\mathrm{p}(\hat{y}_{1:b}\mid\hat{\omega}_k)+\mathrm{p}(\hat{y}_{1:b}\mid\hat{\omega}'_k)}{2K}\right)\cdot\left[\sum_{k=1}^{K}\frac{\mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k)>\tau}\cdot\left(1-\lambda_{\hat{\omega}_k,\hat{y}_{1:b}}\lambda_{\hat{\omega}'_k,\hat{y}_{1:b}}\right)}{K}\right].\end{aligned} \tag{2}$$

The algorithm of Batch-BALanCe and algorithm details are provided in Alg. 2 and §C. The computational complexity analysis is deferred to §C.4.

4. Experiments

In this section, we sought to show the efficacy of BALanCe and Batch-BALanCe on several diverse datasets, including MNIST (LeCun et al., 1998), EMNIST (Cohen et al., 2017) and Fashion-MNIST (Xiao et al., 2017). More experiments on tabular datasets and larger datasets with different metrics, including macro-average AUC, macro-average F1, and NLL, are provided in §F.

4.1 Experimental setup

We split each dataset into unlabeled AL pool $\mathcal{D}_{\text{pool}}$, initial training dataset $\mathcal{D}_{\text{train}}$, validation dataset $\mathcal{D}_{\text{val}}$, test dataset $\mathcal{D}_{\text{test}}$ and unlabeled dataset $\bar{\mathcal{D}}_{\text{pool}}$. $\bar{\mathcal{D}}_{\text{pool}}$ is only used for calculating the Hamming distance between hypotheses and is never used for training BNNs or acquiring labels. At each iteration, we train BNNs with the acquired training dataset and select samples from $\mathcal{D}_{\text{pool}}$ to query labels according to the acquisition function of a chosen algorithm. We provide more experiment details and dataset description in Table 1 and §E.

Besides random selection, BALD and BatchBALD, we also compare BALanCe and Batch-BALanCe with Variation Ratio (Freeman and Freeman, 1965) and Mean STD (Kendall et al., 2015). Note that BALanCe and Batch-BALanCe "activize" BNNs. Therefore, we resort to BALD and Batch-BALD as the natural choice of baseline algorithms, which, to the best of the authors' knowledge, achieve state-of-the-art performance for deep Bayesian AL. Other baselines considered in this paper, including Mean-STD and Variation Ratio, are representative variants of deep Bayesian AL heuristics.

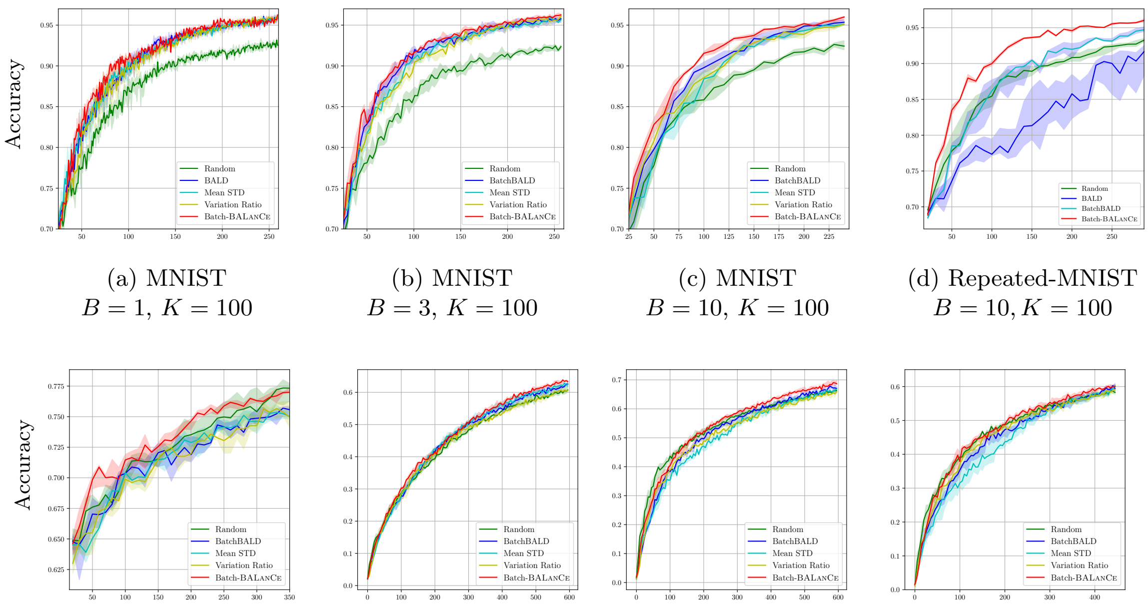
	CINIC10	EMNIST Balanced	EMNIST ByMerge	EMNIST ByClass	RepeatedMNIST	MNIST	HAR	DRIFT	Dry Bean
# classes	10	47	47	62	10	10	6	6	7
# samples	270,000	131,600	814,255	814,255	190,000	70,000	10,299	13,910	13,611
budget	1,400	600	600	450	300	260	200	200	200
τ	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$	$\frac{\text{val err}}{4}$
$\lvert\bar{\mathcal{D}}_{\text{pool}}\rvert$	40,000	18,800	188,000	188,000	10,000	10,000	5,340 (AL pool)	9,898 (AL pool)	9,597 (AL pool)

Table 1: Experiment details on different datasets.

4.2 Batch-mode deep Bayesian AL on balanced dataset

We also compare 5 different models with acquisition sizes $B = 1$, $B = 3$ and $B = 10$ on MNIST dataset. The MC dropout number $K = 100$ for all the methods. The threshold τ for Batch-BALanCe is annealed by setting τ to $\varepsilon/2$ in each AL loop. Note that when $B = 3$, we can compute the acquisition function with all $y_{1:b}$ configurations for $b = 1, 2, 3$. When $b \geq 4$, we approximate the acquisition function with Monte-Carlo sampling. Fig. 1 (a)-(c) show that BALanCe and Batch-BALanCe are consistently better than other baseline methods for MNIST dataset.

We further compare Batch-BALanCe with other baseline methods on three datasets with balanced classes—Repeated-MNIST, Fashion-MNIST and EMNIST-Balanced. The acquisition size B for Repeated-MNIST and Fashion-MNIST is 10 and is 5 for EMNIST-Balanced dataset. The threshold τ of Batch-BALanCe is annealed by setting $\tau = \varepsilon/4$. The learning curves of accuracy are shown in Fig. 1 (d)-(f). For Repeated-MNIST dataset, BALD performs poorly and is worse than random selection. BatchBALD is able to cope with the replication after certain number of AL loops, which is aligned with result showed in Kirsch et al. (2019). The Batch-BALanCe is able to beat all the other methods on this dataset. An ablation study about repetition number and performance can be found in §F.1. For Fashion-MNIST dataset, Batch-BALanCe outperforms random selection but the other methods fail. For EMNIST dataset, Batch-BALanCe is slightly better than BatchBALD.



(a) MNIST $B = 1,\ K = 100$

(b) MNIST $B = 3,\ K = 100$

(c) MNIST $B = 10,\ K = 100$

(d) Repeated-MNIST $B = 10, K = 100$

(e) Fashion-MNIST $B = 10,\ K = 100$

(f) EMNIST-Balanced $B = 5,\ K = 10$

(g) EMNIST-ByMerge $B = 5,\ K = 10$

(h) EMNIST-ByClass $B = 5,\ K = 10$

Figure 1: Experimental results on MNIST, Repeated-MNIST, Fashion-MNIST, EMNIST-Balanced, EMNIST-ByClass and EMNIST-ByMerge dataset. For all plots, the y-axis represents accuracy and x-axis represents the number of queried examples.

4.3 Batch-mode deep Bayesian AL on imbalanced dataset

We further compare different algorithms with two imbalanced datasets: EMNIST-ByMerge and EMNIST-ByClass. The τ for Batch-BALanCe is set $\varepsilon/4$ in each AL loop. $B = 5$ and $K = 10$ for all the methods. As pointed out by Kirsch et al. (2019), BatchBALD performs poorly in imbalanced dataset settings. BALanCe and Batch-BALanCe can cope with the imbalanced data settings. The result is shown in Fig. 1 (g) and (h). Further results on other datasets and under different metrics are provided in §F.

5. Conclusion

We have proposed a novel deep Bayesian active learning framework, with BALanCe for the sequential setting and Batch-BALanCe as the batched-mode extension. BALanCe and Batch-BALanCe employ novel acquisition functions which leverage hypothesis structure captured by equivalence classes without explicitly constructing them. Batch-BALanCe selects a batch of samples at each iteration which can reduce the overhead of retraining the model and save labeling effort. The proposed algorithms achieve state-of-the-art performance on active learning benchmarks both in balanced and imbalanced dataset settings.

Acknowledgments

This work was supported in part by C3.ai DTI Research Award 049755.

References

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 437–442, 2013.

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. In *International Conference on Machine Learning (ICML)*, June 2013.

Yuxin Chen, S. Hamed Hassani, and Andreas Krause. Near-optimal bayesian active learning with correlated and noisy tests, 2016.

Yuxin Chen, Jean-Michel Renders, Morteza Haghir Chehreghani, and Andreas Krause. Efficient online learning for optimizing value of information: Theory and application to interactive troubleshooting. *arXiv preprint arXiv:1703.05452*, 2017.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

Sanjoy Dasgupta and J Langford. Active learning. *Encyclopedia of Machine Learning*, 2011.

Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

Linton C Freeman and Linton C Freeman. *Elementary applied statistics: for students in behavioral science*. New York: Wiley, 1965.

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

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. *arXiv preprint arXiv:1010.3091*, 2010.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Wei-Ning Hsu and Hsuan-Tien Lin. Active learning by learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Jiaji Huang, Rewon Child, Vinay Rao, Hairong Liu, Sanjeev Satheesh, and Adam Coates. Active learning for speech recognition: the power of gradients. *arXiv preprint arXiv:1612.03226*, 2016.

Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. *Advances in neural information processing systems*, 23: 892–900, 2010.

David Janz, Jos van der Westhuizen, and José Miguel Hernández-Lobato. Actively learning what makes a discrete sequence valid. *arXiv preprint arXiv:1708.04465*, 2017.

Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha Srinivasa. Near optimal bayesian active learning for decision making. In *Artificial Intelligence and Statistics*, pages 430–438. PMLR, 2014.

Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298*, 2014.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *arXiv preprint arXiv:1906.08158*, 2019.

Murat Koklu and Ilker Ali Ozkan. Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture*, 174: 105507, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Qiang Liu, Zhaocheng Liu, Xiaofang Zhu, Yeliang Xiu, and Jun Zhu. Deep active learning by model interpretability. *arXiv preprint arXiv:2007.12100*, 2020.

Joaquin Quinonero-Candela, Carl Edward Rasmussen, Fabian Sinz, Olivier Bousquet, and Bernhard Schölkopf. Evaluating predictive uncertainty challenge. In *Machine Learning Challenges Workshop*, pages 1–27. Springer, 2005.

Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder networks. *arXiv preprint arXiv:1507.02672*, 2015.

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

Burr Settles. Active learning. *Synthesis lectures on artificial intelligence and machine learning*, 6(1):1–114, 2012.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.

Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A Ryan, Margie L Homer, and Ramón Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166:320–329, 2012.

Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

Zheng Wang and Jieping Ye. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(3):1–23, 2015.

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

Yifan Yan, Sheng-Jun Huang, Shaoyi Chen, M. Liao, and J. Xu. Active learning with query generation for cost-effective text classification. In *AAAI*, 2020.

Appendix A. Challenges with deep Bayesian AL

A.1 The most informative selection criterion

BALD uses mutual information between the model prediction for each sample and parameters of the model as the acquisition function. It captures the reduction of model uncertainty by receiving a label y of a data point x: $\mathbb{I}\left(y; \omega \mid x, \mathcal{D}_{\text{train}}\right) = \mathbb{H}\left(y \mid x, \mathcal{D}_{\text{train}}\right) - \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})}\left[\mathbb{H}\left(y \mid x, \omega, \mathcal{D}_{\text{train}}\right)\right]$ where $\mathbb{H}$ denotes the Shannon entropy Shannon (1948). Kirsch et al. (2019) further proposed BatchBALD as an extension of BALD whereby the mutual information between a joint of multiple data points and the model parameters is estimated as

$$\Delta_{\text{BatchBALD}}(x_{1:B} \mid \mathcal{D}_{\text{train}}) \triangleq \mathbb{I}(y_{1:B}; \omega \mid x_{1:B}, \mathcal{D}_{\text{train}}).$$

A numerical example illustrating the limitation of BALD BALD can be ineffective when the hypothesis samples are heavily biased and cluttered towards sub-optimal hypotheses. Below, we provide a concrete example where such selection criterion may be undesirable.

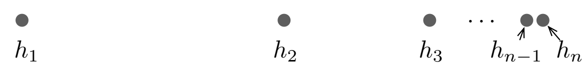


Figure 2: A stylized example where the most informative selection criterion underperforms the equivalence-class-based criterion.

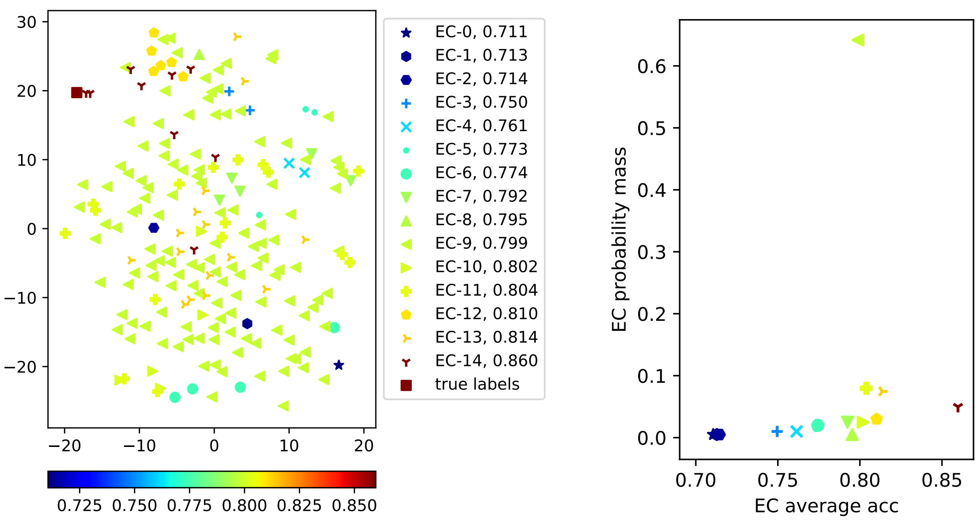
Consider the problem shown in Fig. 2. The hypothesis class $\mathcal{H} = \{h_1, \ldots, h_n\}$ is structured such that

$$d_{\mathrm{H}}(h_i, h_j) = \begin{cases} 2^{1-i} - 2^{1-j} & \text{if } i < j, \\ 2^{1-j} - 2^{1-i} & \text{o.w.} \end{cases}$$

where $d_{\mathrm{H}}(h_i, h_j)$ denotes the fraction of labels h_i and h_j disagree upon when making predictions on *i.i.d.* samples of data points. We further assume that for any subset of hypotheses $S \subseteq \mathcal{H}$, there exists a data point whose label they agree upon. Assume each hypothesis h_i has an equal probability and the target error rate is σ. On the one hand, note that BALD does not consider $d_{\mathrm{H}}(h_i, h_j)$, and therefore on average it requires $\log n$ examples to identify any target hypothesis. On the other hand, to achieve a target error rate of σ, one only needs to differentiate all pairs of hypotheses h_i, h_j of distance $d_{\mathrm{H}}(h_i, h_j) > \sigma$ (i.e., by selecting training examples to rule out at least one of h_i, h_j). Therefore, a "smarter" AL policy could query examples to sequentially check the consistency of $h_1, h_2, \ldots, h_n$ until all remaining hypotheses are within distance σ. It is easy to check that this requires $\log(1/\sigma)$ examples before reaching the error rate σ. The gap between BALD and the above policy $\frac{\log n}{\log(1/\sigma)}$ could be large as n increases.

A.2 Empirical validation: Illustration and additional experiments

An empirical example We now show an empirical example to provide some intuition as to why BALanCe and Batch-BALanCe are effective in practice. In Fig. 3, we demonstrate the potential issues introduced by MC dropout when sampling from a BNN trained on a benchmark multi-classification dataset. Here, the hypothesis samples are grouped into *equivalence classes* (ECs) (Golovin et al., 2010) according to the Hamming distance between their predictions as shown in Fig. 3a. Informally, an equivalence class contains hypotheses that are close in their predictions for a randomly selected set of examples (See §2 for its formal definition). We note from Fig. 3b that the probability mass of the models sampled from the BNN is centered around the mode of the approximate posterior distribution, while little coverage is seen on models of higher accuracy. Consequently, MIS strategy tends to select training examples that reveal the maximal information w.r.t. the sampled distribution, rather than guiding the active learner towards learning high accuracy models.



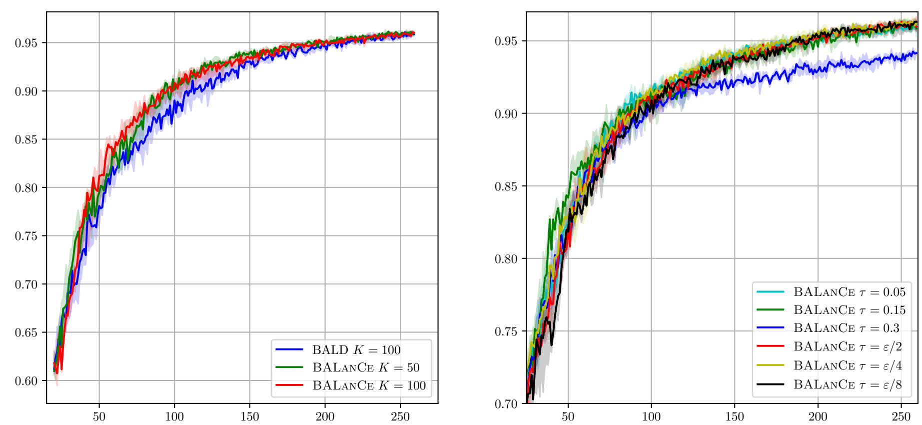
(a)

(b)

Figure 3: (a) Samples from posterior BNN via MC dropout; embedding is generated by applying t-SNE on the disagreement between hypotheses. The colorbar indicates the (approximate) test accuracy of the sampled neural networks on the MNIST dataset. See §A.2 for details of the training and sampling process. (b) Probability mass (y-axis) of equivalence classes (sorted by the average accuracy of the enclosed hypotheses as the x-axis).

The detailed setup for generating the above example is as follows. We train a BNN with an imbalanced MNIST training subset that contains 28 images for each digit in [1-8] and 1 image for digits 0 and 9. The cross-entropy loss is reweighted to balance the training dataset during training. We obtain 200 MC dropout samples of the trained BNN and use them to get the predictions on $\bar{\mathcal{D}}_{\text{pool}}$. We compute the Hamming distances for predictions all sample pairs and use these precomputed distances to plot the predictions with t-SNE (Van der Maaten and Hinton, 2008). The equivalence classes are approximated by farthest-first traversal algorithm (FFT) (Gonzalez, 1985). In Fig. 3, the equivalence classes are highly imbalanced. The ground truth $\bar{\mathcal{D}}_{\text{pool}}$ dataset labels represent the target hypotheses embedding. This figure highlights the scenario where the *equivalence class-based* methods, e.g. ECED and BALanCe are better than MIS and BALD.

Quantitative evaluation We compare BALD and BALanCe with batch size $B = 1$ and different K's on an imbalanced MNIST dataset which is created by removing a random portion of images for each class in the training dataset. Fig. 4 (a) shows that BALanCe performs the best with a large margin to the curve of BALD. Note that BALANCE with $K = 50$ is also better than BALD with $K = 100$.



(a) Acc vs. # samples for different K's. (b) Acc vs. # samples for different τ's.

Figure 4: Learning curves of different K and τ for BALANCE.

We also study the influence of τ for BALANCE on MNIST dataset. Denote the validation error rate of BNN model by ε. BALANCE with fixed $\tau = 0.05, 0.15, 0.3$ and annealing $\tau = \varepsilon/2, \varepsilon/4, \varepsilon/8$ are run on MNIST dataset and the learning curves are shown in Fig. 4 (b). The BALANCE is robust to τ. However, when τ is set 0.3 and the test accuracy gets around 0.88, the accuracy improvement becomes slow. The reason for this slow improvement is that the threshold τ is too large and all the pairs of MC dropout samples are treated as in the same equivalence class and the acquisition functions for all the samples in the AL pool are zeros. In another word, the BALANCE degrades to random selection when τ is too large.

Appendix B. The equivalence-class-based selection criterion

B.1 Equivalence class edge cutting

Consider the problem statement in §2. If $\sigma = 0$ and tests are noise-free, this problem can be solved near-optimally by the *equivalence class edge cutting* (EC^2) algorithm (Golovin et al., 2010). EC^2 employs an edge-cutting strategy based on a weighted graph $G = (\mathcal{H}, \mathcal{E})$, where vertices represent hypotheses and edges link hypotheses that we want to distinguish between. Here $\mathcal{E} \triangleq \{\{h, h'\} : r(h) \neq r(h')\}$ contains all pairs of hypotheses that have different equivalence classes. We define a weight function $W : \mathcal{E} \to \mathbb{R}_{\geq 0}$ by $W(\{h, h'\}) \triangleq \mathrm{p}(h) \cdot \mathrm{p}(h')$. A sample x with label y is said to "cut" an edge, if at least one hypothesis is inconsistent with y. Denote $\mathcal{E}(x, y) \triangleq \{\{h, h'\} \in \mathcal{E} : \mathrm{p}(y \mid x, h) = 0 \vee \mathrm{p}(y \mid x, h') = 0\}$ as the set of edges cut by labeling x as y. The EC^2 objective is then defined as the total weight of edges cut by the current $\mathcal{D}_{\text{train}}$: $f_{\mathrm{EC}^2}(\mathcal{D}_{\text{train}}) \triangleq W\left(\bigcup_{(x,y)\in\mathcal{D}_{\text{train}}} \mathcal{E}(x, y)\right)$. EC^2 algorithm greedily maximizes this objective per iteration. The acquisition function for EC^2 is

$$\Delta_{\mathrm{EC}^2}\left(x \mid \mathcal{D}_{\text{train}}\right) \triangleq \mathbb{E}_y\left[f\left(\mathcal{D}_{\text{train}} \cup \{(x, y)\}\right) - f(\mathcal{D}_{\text{train}}) \mid \mathcal{D}_{\text{train}}\right].$$

B.2 Equivalence class edge discounting algorithm

The acquisition function of *Equivalence Class Edge Discounting* algorithm (ECED) (Chen et al., 2016) takes undesired contribution by noise into account. Given a data point and its label (x, y), ECED discounts all model parameters by their likelihood ratio: $\lambda_{h,y} \triangleq \frac{\mathrm{p}(y|h,x)}{\max_{y'} \mathrm{p}(y'|h,x)}$. After we get $\mathcal{D}_{\text{train}}$, the value of assigning label y to a data point x is defined as the total amount of edge weight discounted: $\delta(x, y \mid \mathcal{D}_{\text{train}}) \triangleq \sum_{\{h,h'\}\in\mathcal{E}} \mathrm{p}(h, \mathcal{D}_{\text{train}})\mathrm{p}(h', \mathcal{D}_{\text{train}}) \cdot (1 - \lambda_{h,y}\lambda_{h',y})$, where $\mathcal{E} = \{\{h, h'\} : r(h) \neq r(h')\}$ consists of all unordered pairs of hypothesis corresponding to different equivalence classes. Further, ECED augments the above value function δ with an offset value such that the value of a non-informative test is 0. The offset value of labeling x as label y is defined as: $\nu(x, y \mid \mathcal{D}_{\text{train}}) \triangleq \sum_{\{h,h'\}\in\mathcal{E}} \mathrm{p}(h, \mathcal{D}_{\text{train}})\mathrm{p}(h', \mathcal{D}_{\text{train}}) \cdot (1 - \max_h \lambda^2_{h,y})$. The overall acquisition function function of ECED is:

$$\Delta_{\mathrm{ECED}}(x \mid \mathcal{D}_{\text{train}}) \triangleq \mathbb{E}_y\left[\delta(x, y \mid \mathcal{D}_{\text{train}}) - \nu(x, y \mid \mathcal{D}_{\text{train}})\right]. \tag{3}$$

Limitation of the ECED **algorithm** ECED is not directly applicable to deep Bayesian AL tasks, since computing the acquisition function in Eq. (3) needs to integrate over the hypotheses space, which is intractable for large models (such as deep BNN). Chen et al. (2017) approximate the acquisition function by *dynamic hypothesis enumeration*, which enumerates the hypotheses in decreasing order of probability. Unfortunately, such an enumeration strategy is still infeasible for BNNs. Moreover, it is nontrivial to extend to batch-mode setting since the number of possible candidate batches and the number of labels for the candidate batch grow exponentially with the batch size. Therefore, we need more efficient approaches to approximate the ECED acquisition function when dealing with BNNs in both fully sequential setting and batch-mode setting.

Appendix C. Algorithmic details

C.1 Derivation of acquisition functions of BALanCe and Batch-BALanCe

In each AL loop, the ECED algorithm selects a sample from AL pool according to the acquisition function

$$\Delta_{\mathrm{ECED}}(x \mid \mathcal{D}_{\text{train}}) \triangleq \mathbb{E}_y\left[\sum_{\{\omega,\omega'\}\in\mathcal{E}} W_{\omega,\omega'}\left(1 - \lambda_{\omega,y}\lambda_{\omega',y} - \left(1 - \max_{\omega} \lambda^2_{\omega,y}\right)\right)\right],$$

where $\mathcal{E}$ is the total edges with adjacent nodes in different equivalence classes and $\lambda_{\omega,y} = \frac{\mathrm{p}(y|\omega)}{\max_{y'} \mathrm{p}(y'|\omega)}$. $W_{\omega,\omega'}$ is the weight for edge $\{\omega, \omega'\}$ which is maintained by ECED algorithm. After we observe y of selected x, we update the weights of all edges with $W_{\omega,\omega'} = W_{\omega,\omega'} \cdot \mathrm{p}(y \mid \omega)\mathrm{p}(y \mid \omega')$. In the deep Bayesian AL setting, the offset term $1 - \max_\omega \lambda^2_{\omega,y}$ can be removed when we use deep BNN. However, we can not enumerate all $\{\omega, \omega'\} \in \mathcal{E}$ in this setting since there are an infinite number of hypotheses in the hypothesis space. Moreover, we can not even estimate the acquisition function of ECED on a subset of sampled hypotheses by MC dropouts since building equivalence classes with best ϵ is NP-hard.

If we sample $\{\omega, \omega'\}$ according to posterior $\mathrm{p}(\omega \mid \mathcal{D}_{\mathrm{train}})$ and check whether $\{\omega, \omega'\} \in \hat{\mathcal{E}}$ by Hamming distance in the way we describe in §3.1, we will get

$$\begin{aligned}\Delta_{\mathrm{ECED}}(x \mid \mathcal{D}_{\mathrm{train}}) &\approx \mathbb{E}_y \left[\sum_{\{\omega,\omega'\}\in\mathcal{E}} W_{\omega,\omega'} \left(1 - \lambda_{\omega,y}\lambda_{\omega',y}\right) \right] \\ &\approx \mathbb{E}_y \left[\mathbb{E}_{\omega,\omega'\sim \mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega,\omega')>\tau} \cdot \frac{W_{\omega,\omega'}}{\mathrm{p}(\omega \mid \mathcal{D}_{\mathrm{train}})\mathrm{p}(\omega' \mid \mathcal{D}_{\mathrm{train}})} \cdot \left(1 - \lambda_{\omega,y}\lambda_{\omega',y}\right) \right] \\ &\propto \mathbb{E}_y \left[\mathbb{E}_{\omega,\omega'\sim \mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega,\omega')>\tau} \cdot \left(1 - \lambda_{\omega,y}\lambda_{\omega',y}\right) \right].\end{aligned}$$

Inspired by the weight discounting mechanism of ECED, we define the acquisition function of BALanCe $\Delta_{\mathrm{BALanCe}}(x \mid \mathcal{D}_{\mathrm{train}})$ as

$$\Delta_{\mathrm{BALanCe}}(x \mid \mathcal{D}_{\mathrm{train}}) \triangleq \mathbb{E}_y \left[\mathbb{E}_{\omega,\omega'\sim \mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega,\omega')>\tau} \cdot \left(1 - \lambda_{\omega,y}\lambda_{\omega',y}\right) \right].$$

After we get K pairs of MC dropouts, the acquisition function $\Delta_{\mathrm{BALanCe}}(x \mid \mathcal{D}_{\mathrm{train}})$ can be approximated as follows:

$$\begin{aligned}\Delta_{\mathrm{BALanCe}}(x \mid \mathcal{D}_{\mathrm{train}}) &= \mathbb{E}_{\mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{E}_{\mathrm{p}(y\mid\omega)} \left[\mathbb{E}_{\omega,\omega'\sim \mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega,\omega')>\tau} \left(1 - \lambda_{\omega,y}\lambda_{\omega',y}\right) \right] \\ &\approx \sum_{\hat{y}} \left(\frac{1}{2K} \sum_{k=1}^{K} \mathrm{p}(\hat{y} \mid \hat{\omega}_k) + \mathrm{p}(\hat{y} \mid \hat{\omega}'_k) \right) \left[\frac{1}{K} \sum_{k=1}^{K} \mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k)>\tau} \left(1 - \lambda_{\hat{\omega}_k,\hat{y}}\lambda_{\hat{\omega}'_k,\hat{y}}\right) \right].\end{aligned}$$

In batch-mode setting, the acquisition function of Batch-BALanCe for a batch $x_{1:b}$ is

$$\Delta_{\mathrm{Batch-BALanCe}}(x_{1:b} \mid \mathcal{D}_{\mathrm{train}}) \triangleq \mathbb{E}_{y_{1:b}} \left[\mathbb{E}_{\omega,\omega'\sim \mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega,\omega')>\tau} \cdot \left(1 - \lambda_{\omega,y_{1:b}}\lambda_{\omega',y_{1:b}}\right) \right].$$

Similar to the fully sequential setting, we can approximate $\Delta_{\mathrm{Batch-BALanCe}}(x_{1:b} \mid \mathcal{D}_{\mathrm{train}})$ with K pairs of MC dropouts. The $x_{1:b}$ are chosen in a greedy manner. For iteration b inside a batch, the $x_{1:b-1}$ are fixed and x_b is selected according to

$$\begin{aligned}&\Delta_{\mathrm{Batch-BALanCe}}(x_{1:b} \mid \mathcal{D}_{\mathrm{train}}) \\ =&\mathbb{E}_{\mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{E}_{\mathrm{p}(y_{1:b}\mid\omega)} \left[\mathbb{E}_{\omega,\omega'\sim \mathrm{p}(\omega\mid\mathcal{D}_{\mathrm{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega,\omega')>\tau} \left(1 - \lambda_{\omega,y_{1:b}}\lambda_{\omega',y_{1:b}}\right) \right] \\ \approx&\sum_{\hat{y}_{1:b}} \left(\frac{1}{2K} \sum_{k=1}^{K} \mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}_k) + \mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}'_k) \right) \left[\frac{1}{K} \sum_{k=1}^{K} \mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k)>\tau} \left(1 - \lambda_{\hat{\omega}_k,\hat{y}_{1:b}}\lambda_{\hat{\omega}'_k,\hat{y}_{1:b}}\right) \right].\end{aligned}$$

C.2 Efficient implementation for batch-mode setting

In Alg. 2, we can store $\mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}_k)$ in a matrix $\hat{P}_{1:b-1}$ and $\mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}'_k)$ in matrix $\hat{P}_{1:b-1}$ for iteration $b-1$. The shape of $\hat{P}_{1:b-1}$ and $\hat{P}'_{1:b-1}$ is $K \times C^{b-1}$. $\mathrm{p}(\hat{y}_b \mid \hat{\omega}_k)$ can be stored in $\hat{P}_b$ and $\mathrm{p}(\hat{y}_b \mid \hat{\omega}'_k)$ in $\hat{P}'_b$. The shape of $\hat{P}_b$ and $\hat{P}'_b$ is $K \times C$. Then, we compute probability of $\mathrm{p}(\hat{y}_{1:b})$ as follows:

$$\begin{aligned}
\mathrm{p}(\hat{y}_{1:b}) =& \frac{1}{2K}\sum_{k=1}^{K}\mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}_k) + \mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}'_k) \\
=& \frac{1}{2K}\sum_{k=1}^{K}\mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}_k)\mathrm{p}(\hat{y}_b \mid \hat{\omega}_k) + \mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}'_k)\mathrm{p}(\hat{y}_b \mid \hat{\omega}'_k) \\
=& \frac{1}{2K}(\hat{P}_{1:b-1}^{\top}\hat{P}_b + \hat{P}'^{\top}_{1:b-1}\hat{P}'_b).
\end{aligned}$$

The $\hat{P}_{1:b-1}^{\top}\hat{P}_b$ and $\hat{P}'^{\top}_{1:b-1}\hat{P}'_b$ can be flattened to shape $1 \times C^b$ after matrix multiplication. We store $\max_{\hat{y}_{1:b-1}} \mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}_k)$ in a matrix $\hat{A}_{1:b-1}$ and $\max_{\hat{y}'_{1:b-1}} \mathrm{p}(\hat{y}'_{1:b-1} \mid \hat{\omega}'_k)$ in a matrix $\hat{A}'_{1:b-1}$. The shape of $\hat{A}_{1:b-1}$ and $\hat{A}'_{1:b-1}$ is $K \times 1$. We can compute $\lambda_{\hat{\omega},\hat{y}_{1:b}}$ inside edge weight discount expression by

$$\begin{aligned}
\hat{A}_{1:b} &= \hat{A}_{1:b-1} \odot \max_{\hat{y}_b} \hat{P}_b; \\
\mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}_k) &= \mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}_k)\mathrm{p}(\hat{y}_b \mid \hat{\omega}_k) = \hat{P}_{1:b-1} \otimes \hat{P}_b; \\
\lambda_{\hat{\omega},\hat{y}_{1:b}} &= \frac{\mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}_k)}{\max_{\hat{y}_{1:b}} \mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}_k)} = \frac{\hat{P}_{1:b-1} \otimes \hat{P}_b}{\hat{A}_{1:b}}.
\end{aligned}$$

$\odot$ is element-wise matrix multiplication and $\otimes$ is the outer-product operator along the first dimension. After the outer product operation, we can reshape the matrix by flattening all the dimensions after 1st dimension to maintain consistency. Similarly, we can compute $\hat{A}'_{1:b}$, $\mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}'_k)$ and $\lambda_{\hat{\omega}',\hat{y}_{1:b}}$ with matrix operations. The indicator function $\mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k)>\tau}$ can be stored in a matrix with shape $K \times 1$. The acquisition function can be computed with all matrix operations as follows:

$$\begin{aligned}
&\Delta_{\text{Batch-BALANCE}}(x_{1:b} \mid \mathcal{D}_{\text{train}}) \\
=&\mathbb{E}_{\mathrm{p}(\omega|\mathcal{D}_{\text{train}})}\mathbb{E}_{\mathrm{p}(y_{1:b}|\omega)}\left[\mathbb{E}_{\omega,\omega'\sim \mathrm{p}(\omega|\mathcal{D}_{\text{train}})}\mathbb{1}_{d_{\mathrm{H}}(\omega,\omega')>\tau}\left(1 - \lambda_{\omega,y_{1:b}}\lambda_{\omega',y_{1:b}}\right)\right] \\
\approx&\sum_{\hat{y}_{1:b}}\left(\frac{1}{2K}\sum_{k=1}^{K}\mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}_k) + \mathrm{p}(\hat{y}_{1:b} \mid \hat{\omega}'_k)\right)\left[\frac{1}{K}\sum_{k=1}^{K}\mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k,\hat{\omega}'_k)>\tau}\left(1 - \lambda_{\hat{\omega}_k,\hat{y}_{1:b}}\lambda_{\hat{\omega}'_k,\hat{y}_{1:b}}\right)\right] \\
=&\left(\frac{1}{K}\mathbb{1}_{D(\hat{\omega}_k,\hat{\omega}'_k)>\tau}\right)^{\top}\left(1 - \frac{\hat{P}_{1:b-1} \otimes \hat{P}_b}{\hat{A}_{1:b}} \odot \frac{\hat{P}'_{1:b-1} \otimes \hat{P}'_b}{\hat{A}'_{1:b}}\right)\left[\frac{1}{2K}(\hat{P}_{1:b-1}^{\top}\hat{P}_b + \hat{P}'^{\top}_{1:b-1}\hat{P}'_b)\right]^{\top}.
\end{aligned}$$

C.3 Importance sampling of configurations

Given that $\mathrm{p}(y_{1:b} \mid \omega)$ can be factorized as $\mathrm{p}(y_{1:b-1} \mid \omega) \cdot \mathrm{p}(y_b \mid \omega)$, the acquisition function can be written as:

$$\begin{aligned}
&\Delta_{\text{Batch-BALANCE}}(x_{1:b} \mid \mathcal{D}_{\text{train}}) \\
\triangleq &\mathbb{E}_{y_{1:b}}\left[\mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega_k, \omega'_k) > \tau} \left(1 - \lambda_{\omega, y_{1:b}} \lambda_{\omega', y_{1:b}}\right)\right] \\
= &\mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{E}_{\mathrm{p}(y_{1:b} \mid \omega)} \left[\mathbb{E}_{\omega, \omega' \sim \mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega_k, \omega'_k) > \tau} \left(1 - \lambda_{\omega, y_{1:b}} \lambda_{\omega', y_{1:b}}\right)\right] \\
= &\mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{E}_{\mathrm{p}(y_{1:b-1} \mid \omega)} \mathbb{E}_{\mathrm{p}(y_b \mid \omega)} \left[\mathbb{E}_{\omega, \omega' \sim \mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega_k, \omega'_k) > \tau} \left(1 - \lambda_{\omega, y_{1:b}} \lambda_{\omega', y_{1:b}}\right)\right]
\end{aligned}$$

Suppose we have M samples of $y_{1:b-1}$ from $\mathrm{p}(y_{1:b-1})$, we perform importance sampling using $\mathrm{p}(y_{1:b-1})$ to estimate the acquisition function:

$$\begin{aligned}
&\Delta_{\text{Batch-BALANCE}}(x_{1:b} \mid \mathcal{D}_{\text{train}}) \\
= &\mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{E}_{\mathrm{p}(y_{1:b-1})} \frac{\mathrm{p}(y_{1:b-1} \mid \omega)}{\mathrm{p}(y_{1:b-1})} \mathbb{E}_{\mathrm{p}(y_b \mid \omega)} \left[\mathbb{E}_{\omega, \omega' \sim \mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega, \omega') > \tau} \left(1 - \lambda_{\omega, y_{1:b}} \lambda_{\omega', y_{1:b}}\right)\right] \\
= &\mathbb{E}_{\mathrm{p}(y_{1:b-1})} \mathbb{E}_{\mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{E}_{\mathrm{p}(y_b \mid \omega)} \frac{\mathrm{p}(y_{1:b-1} \mid \omega)}{\mathrm{p}(y_{1:b-1})} \left[\mathbb{E}_{\omega, \omega' \sim \mathrm{p}(\omega \mid \mathcal{D}_{\text{train}})} \mathbb{1}_{d_{\mathrm{H}}(\omega, \omega') > \tau} \left(1 - \lambda_{\omega, y_{1:b}} \lambda_{\omega', y_{1:b}}\right)\right] \\
\approx &\frac{1}{M} \sum_{\hat{y}_{1:b-1}}^{M} \sum_{\hat{y}_b} \frac{\frac{1}{K} \sum_{k=1}^{K} \mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}_k) \mathrm{p}(\hat{y}_b \mid \hat{\omega}_k) + \mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}'_k) \mathrm{p}(\hat{y}_b \mid \hat{\omega}'_k)}{\mathrm{p}(\hat{y}_{1:b-1})} \cdot \\
&\qquad \left[\frac{1}{K} \sum_{k=1}^{K} \mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k, \hat{\omega}'_k) > \tau} \left(1 - \lambda_{\hat{\omega}_k, \hat{y}_{1:b}} \lambda_{\hat{\omega}'_k, \hat{y}_{1:b}}\right)\right] \\
= &\left(\frac{1}{K} \mathbb{1}_{d_{\mathrm{H}}(\hat{\omega}_k, \hat{\omega}'_k) > \tau}\right)^{\top} \left(1 - \frac{\hat{P}_{1:b-1} \otimes \hat{P}_b}{\hat{A}_{1:b}} \odot \frac{\hat{P}'_{1:b-1} \otimes \hat{P}'_b}{\hat{A}'_{1:b}}\right) \left(\frac{1}{M} \frac{\hat{P}_{1:b-1}^{\top} \hat{P}_b + \hat{P}'^{\top}_{1:b-1} \hat{P}'_b}{\mathbb{1}^{\top} \left(\hat{P}_{1:b-1} + \hat{P}'_{1:b-1}\right)}\right)^{\top}.
\end{aligned}$$

Here we save $\mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}_k)$ and $\mathrm{p}(\hat{y}_{1:b-1} \mid \hat{\omega}'_k)$ for M samples in $\hat{P}_{1:b-1}$ and $\hat{P}'_{1:b-1}$. The shape of $\hat{P}_{1:b-1}$ and $\hat{P}'_{1:b-1}$ is $K \times M$. $\odot$ is element-wise matrix multiplication and $\otimes$ is the outer-product operator along first dimension. After the outer product operation, we can reshape the matrix by flattening all the dimensions after the 1st dimension. $\mathbb{1}$ is a matrix of 1s with shape $K \times 1$. $\hat{P}_{1:b-1}^{\top} \hat{P}_b$ and $\hat{P}'^{\top}_{1:b-1} \hat{P}'_b$ are of shape $M \times C$ and their sum is reshape to $1 \times MC$ after divided by $\mathbb{1}^{\top} \left(\hat{P}_{1:b-1} + \hat{P}'_{1:b-1}\right)$.

C.4 Computational complexity

Let C be the number of classes, B be the acquisition size, K be the MC dropout number and M be the sample number for $y_{1:b}$ configurations. When we compute $\Delta_{\text{Batch-BALANCE}}(x_{1:b} \mid \mathcal{D}_{\text{train}})$ by enumerating all $y_{1:b}$ configurations, the computational complexity of Batch-BALANCE is $\mathcal{O}\left(|\mathcal{D}_{\text{pool}}| \cdot B \cdot C^B \cdot 2K\right)$. The computational complexity of Batch-BALANCE by Monte-Carlo sampling is $\mathcal{O}\left(|\mathcal{D}_{\text{pool}}| \cdot B \cdot \min\{C^B, M\} \cdot 2K\right)$ and it is similar to the one of Batch-BALD. Besides, our algorithms need to iterate over $\bar{\mathcal{D}}_{\text{pool}}$ once to determine the pair distances and $\bar{\mathcal{D}}_{\text{pool}}$ is very small compared to $\mathcal{D}_{\text{pool}}$.

Algorithm 1 Active selection w/ BALANCE

input: $\mathcal{D}_{\text{pool}}$,$\bar{\mathcal{D}}_{\text{pool}}$, a trained BNN and threshold τ
draw K pairs of MC dropout samples $\{\hat{\omega}_k, \hat{\omega}'_k\}_{k=1}^K$ from the trained BNN
for $k \in [K]$ **do**
  calculate the indicator function $\mathbb{1}_{d_{\text{H}}(\hat{\omega}_k, \hat{\omega}'_k) > \tau}$ with predictions on $\bar{\mathcal{D}}_{\text{pool}}$
for each $x \in \mathcal{D}_{\text{pool}}$ **do**
  $\Delta(x) = 0$
  for each $\hat{y}$ **do**
    $\mathrm{p}(\hat{y}) = \frac{1}{2K} \sum_{k=1}^K (\mathrm{p}(\hat{y} \mid \hat{\omega}_k) + \mathrm{p}(\hat{y} \mid \hat{\omega}'_k))$
    $\delta(x, \hat{y}) = 0$
    for $k \in [K]$ **do**
      if $\mathbb{1}_{d_{\text{H}}(\hat{\omega}_k, \hat{\omega}'_k) > \tau} = 1$ **then**
        $\delta(x, \hat{y}) \leftarrow \delta(x, \hat{y}) + \frac{1}{K} \left(1 - \lambda_{\hat{\omega}_k, \hat{y}} \lambda_{\hat{\omega}'_k, \hat{y}}\right)$
    $\Delta(x) \leftarrow \Delta(x) + \mathrm{p}(\hat{y}) \cdot \delta(x, \hat{y})$
$x \leftarrow \arg\max_{x \in \mathcal{D}_{\text{pool}}} \Delta(x)$
output: $\{x\}$

Algorithm 2 Batch selection w/ Batch-BALANCE

input: acquisition size B, $\mathcal{D}_{\text{pool}}$ and $\bar{\mathcal{D}}_{\text{pool}}$, trained BNN and threshold τ
$\mathcal{A}_0 = \emptyset$
for $b \in [B]$ **do**
  for $x \in \mathcal{D}_{\text{pool}} \backslash \mathcal{A}_{b-1}$ **do**
    $s_x \leftarrow \Delta_{\text{Batch-BALANCE}}(\mathcal{A}_{b-1} \bigcup \{x\})$
  $x_b \leftarrow \arg\max_{x \in \mathcal{D}_{\text{pool}} \backslash \mathcal{A}_{b-1}} s_x$
  $\mathcal{A}_b \leftarrow \mathcal{A}_{b-1} \bigcup \{x_b\}$
output: batch $\mathcal{A}_B = \{x_1, \ldots, x_B\}$

Appendix D. Related work

Batch-mode Bayesian AL Batch-mode Bayesian AL has shown promising performance for practical AL tasks. Some methods (Gal et al., 2017) choose a batch of samples with top acquisition functions. These methods could lead to performance drops compared to the methods that choose one sample with the highest acquisition function value since these methods will choose similar and correlated samples inside each batch. Kirsch et al. (2019) extended Houlsby et al. (2011) and proposed a batch-mode deep Bayesian AL algorithm, namely BatchBALD. Chen and Krause (2013) formalized a class of interactive optimization problems as adaptive submodular optimization problems and prove a greedy batch-mode approach to these problems is near-optimal as compared to the optimal batch selection policy.

Semi-supervised learning Semi-supervised learning leverages unlabeled data (together with labeled examples) in the training process (Kingma et al., 2014; Rasmus et al., 2015). Some work has combined AL and semi-supervised learning (Wang et al., 2016; Sener and Savarese, 2017; Sinha et al., 2019). Our methods are different from these methods since our methods never leverage unlabeled data to train the models, but rather use the unlabeled pool to inform the selection of data points for AL.

Appendix E. Experiment setup and dataset description

The training set is small in the early stage and could be imbalanced. To avoid overfitting, we train the BNNs at each iteration with early stopping, where we terminate the training of BNNs with patience of 3 epochs. The BNN with the highest validation accuracy is picked and used to calculate the acquisition functions. Additionally, we use weighted cross-entropy loss for training the BNN to mitigate the bias introduced by imbalanced training data. The BNN models are reinitialized in each AL iteration similar to Gal et al. (2017); Kirsch et al. (2019). It decorrelates subsequent acquisitions as the final model performance is dependent on a particular initialization. We use Adam optimizer (Kingma and Ba, 2017) for all the models in the experiments.

In the batch-mode setting, if $b < 4$, Batch-BALanCe enumerates all $y_{1:b}$ configurations to compute the acquisition function $\Delta_{\text{Batch-BALanCe}}$ according to Eq. (2); otherwise, it will use $M = 10,000$ Monte-Carlo samples of $y_{1:b}$ and importance sampling to estimate $\Delta_{\text{Batch-BALanCe}}$. All our results report the median of 6 trials, with lower and upper quartiles. We use these quartiles to draw the filled error bars on our figures.

MNIST We randomly split MNIST training dataset into $\mathcal{D}_{\text{val}}$ with 10,000 samples, $\bar{\mathcal{D}}_{\text{pool}}$ with 10,000 samples and $\mathcal{D}_{\text{pool}}$ with the rest. The initial training dataset contains 20 samples with 2 samples in each class chosen from the AL pool. The BNN model architecture is similar to Kirsch et al. (2019). It consists of two blocks of [convolution, dropout, max-pooling, relu] followed by a two-layer MLP that a two-layer MLP and one dropout between the two layers. The dropout probability is 0.5 in the dropout layers.

Repeated-MNIST Kirsch et al. (2019) show that applying BALD to a dataset that contains many (near) replicated data points leads to poor performance. We again randomly split the MNIST training dataset similar to the settings used on MNIST dataset. We replicate all the samples in AL pool two times and add isotropic Gaussian noise with a standard deviation of 0.1 after normalizing the dataset. The BNN architecture is the same as the one used on MNIST dataset.

EMNIST We further consider the EMNIST dataset under 3 different settings: EMNIST-Balanced, EMNIST-ByClass, and EMNIST-ByMerge. The EMNIST-Balanced contains 47 classes with balanced digits and letters. EMNIST-ByMerge includes digits and letters for a total of 47 unbalanced classes. EMNIST-ByClass represents the most useful organization for classification as it contains the segmented digits and characters for 62 classes comprising [0-9],[a-z], and [A-Z]. We randomly split the training set into $\mathcal{D}_{\text{val}}$ with 18,800 images, $\bar{\mathcal{D}}_{\text{pool}}$ with 18,800 images and $\mathcal{D}_{\text{pool}}$ with the rest of the samples. Similar to Kirsch et al. (2019), we do not use an initial dataset and instead perform the initial acquisition step with the randomly initialized model. The model architecture contains three blocks of [convolution, dropout, max-pooling, relu], with 32, 64, and 128 3x3 convolution filters and 2x2 max pooling. We add a two-layer MLP following the three blocks. 4 dropout layers in total are in each block and MLP with dropout probability 0.5.

Fashion-MNIST Fashion-MNIST is a dataset of Zalando's article images that consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. We randomly split Fashion-MNIST training dataset into $\mathcal{D}_{\text{val}}$ with 10,000 samples, $\bar{\mathcal{D}}_{\text{pool}}$ with 10,000 samples, and $\mathcal{D}_{\text{pool}}$ with the rest of samples. We obtain the initial training dataset that contains 20 samples with 2 samples in each class randomly chosen from the AL pool. The model architecture is similar to the one used on EMNIST dataset with 10 units in the last MLP.

Appendix F. Supplemental experiments

In this section, we provide additional experimental details and supplemental results to demonstrate the competing algorithms under an additional evaluation metric.

F.1 Experiments on other datasets

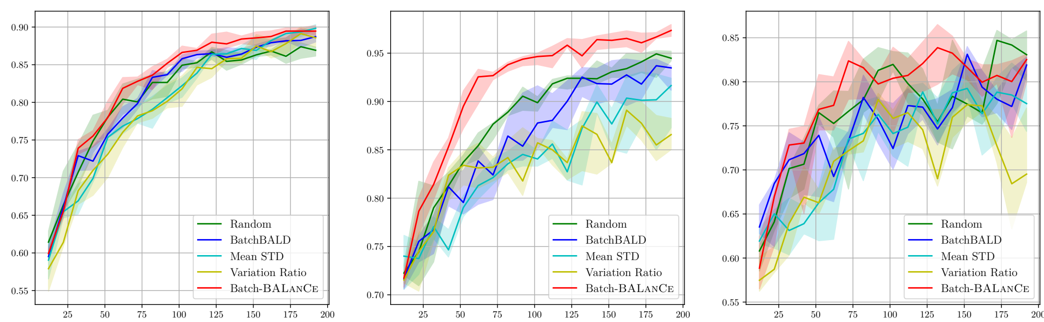
We compare different AL algorithms on tabular datasets including Human Activity Recognition Using Smartphones Data Set (Anguita et al., 2013) (HAR), Gas Sensor Array Drift (Vergara et al., 2012) (DRIFT), and Dry Bean Dataset (Koklu and Ozkan, 2020), as well as a more difficult dataset CINIC-10 (Darlow et al., 2018).

HAR, DRIFT and Dry Bean Dataset We run 6 AL trials for each dataset and algorithm. In each iteration, the BNNs are trained with a learning rate of 0.01 and patience equal to 3 epochs. The BNNs all contain three-layer MLP with ReLU activation and dropout layers in between. The datasets are all split into starting training set, validation set, testing set, and AL pool. The AL pool is also used as $\bar{\mathcal{D}}_{\text{pool}}$. The τ for Batch-BALanCe is set $\varepsilon/4$ in each AL loop. See Table 2 for more experiment details of these 3 datasets.

The learning curves of all 5 algorithms on these 3 tabular datasets are shown in Fig. 5. Batch-BALanCe outperforms all the other algorithms for these 3 datasets. For HAR dataset, both Batch-BALanCe and BatchBALD work better than random selection. In Fig. 5 (b) and (c), Mean STD, Variation Ratio and BatchBALD perform worse than random selection. We find similar effect for some other imbalanced datasets.

dataset	val set size	test set size	hidden unit #	sample # per epoch	K	B
HAR	2K	2,947	(64,64)	4,096	20	10
DRIFT	2K	2K	(32,32)	4,096	20	10
Dry Bean	2K	2K	(8,8)	8,192	20	10

Table 2: Experment details for HAR, DRIFT and Dry Bean Dataset



(a) HAR (b) DRIFT (c) Dry Bean Dataset

Figure 5: Experimental results on 3 tabular datasets. For all plots, the y-axis represents accuracy and x-axis represents the number of queried examples.

CINIC-10 CINIC-10 is a large dataset with 270K images from two sources: CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (Rasmus et al., 2015). The training set is split into an AL pool with 120K samples, 40K $\bar{\mathcal{D}}_{\text{pool}}$ samples, 20K validation samples and 200 starting training samples with 20 samples in each class. We use VGG-11 as the BNN. The number of sampled MC dropout pairs is 50 and the acquisition size is 10. We run 6 trials for this experiment. The learning curves of 5 algorithms are shown in Fig. 6. We can see from Fig. 6 that Batch-BALanCe performs better than all the other algorithms by a large margin in this setting.

Repeated-MNIST with different amounts of repetitions In order to show the effect of redundant data points on BathBALD and Batch-BALanCe, we ran experiments on Repeated-MNIST with an increasing number of repetitions. The learning curves of accuracy for Repeated-MNIST with different repetition numbers can be seen in Fig. 7. A detailed model accuracy on the test dataset when acquired training dataset size is 130 is shown in Table 3. Even though Batch-BALanCe can improve data efficiency (Kirsch et al., 2019), there are still large gaps between learning curves of Batch-BALD and Batch-BALanCe and the gaps become larger when the number of repetitions increases.

F.2 Additional evaluation metrics

Besides accuracy, we compared macro-average AUC, macro-average F1, and NLL for 5 different methods on EMNIST-Balanced and EMNIST-ByMerge datasets in Fig. 8. The acquisition size for all the AL algorithms is 5. Batch-BALanCe is annealed by setting $\tau = \varepsilon/4$. A macro-average AUC computes the AUC independently for each class and then

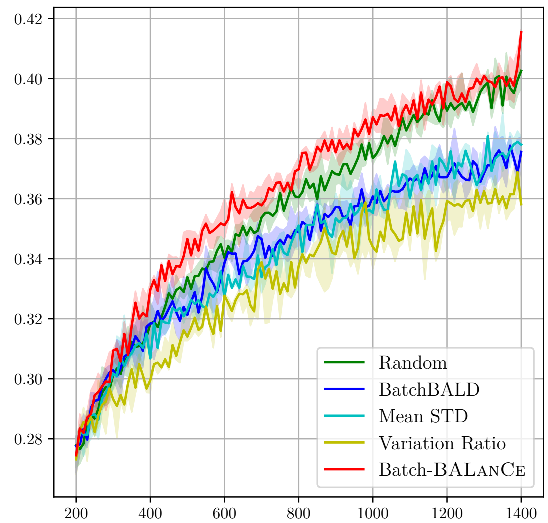


Figure 6: Accuracy vs. # queries on the CINIC-10 dataset.

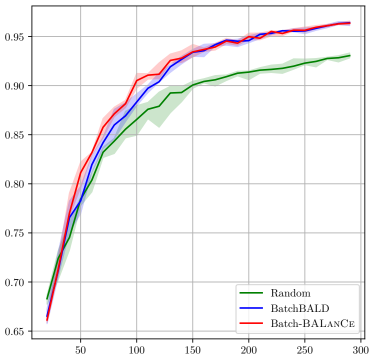
takes the average. Both macro-average AUC and macro-average F1 take class imbalance into account. As shown in Fig. 8, Batch-BALanCe attains better data efficiency compared with baseline models on both balanced and imbalanced datasets.

We also evaluated the negative log-likelihood (NLL) for different AL algorithms. NLL is a popular metric for evaluating predictive uncertainty (Quinonero-Candela et al., 2005). As shown in Fig. 8, Batch-BALanCe maintains a better or comparable quality of predictive uncertainty over test data.

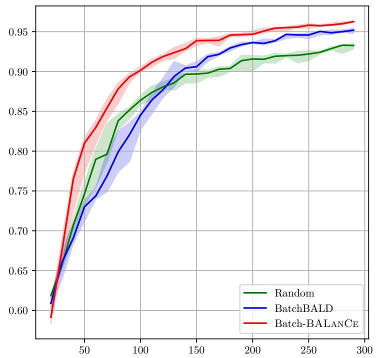
F.3 BALanCe through explicit partitioning over the hypothesis samples (BALanCe-Partition)

Another way of estimating the acquisition function is to construct the equivalence classes explicitly first (e.g. by partitioning the hypothesis spaces into k Voronoi cells via max-diameter clustering and calculate the weight discounts of edges that connect different equivalence classes. Intuitively, explicitly constructing equivalence classes may introduce unnecessary edges as two closeby hypotheses can be partitioned into different equivalence classes; therefore leading to an overestimate of the edge weight discounted. We call this algorithm BALanCe-Partition.

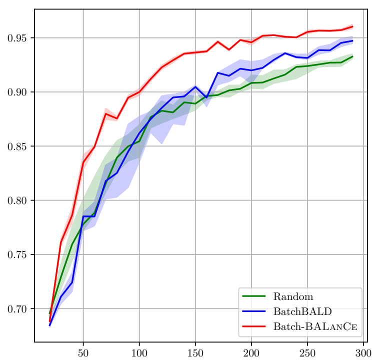
In order to compare with BALanCe and Batch-BALanCe, we sampled K pairs of MC dropouts to estimate the acquisition function of BALanCe-Partition. All the representations of $2K$ MC dropouts on $\bar{\mathcal{D}}_{\text{pool}}$ are generated. We run FFT (Gonzalez, 1985) with Hamming distances and threshold τ on these representations to get approximated ECs. Each data point has at most τ Hamming distance to the corresponding cluster center. FFT is a 2-approx algorithm and the optimal solution with the same cluster number has cluster diameter $\geq \frac{\tau}{2}$. After equivalence classes are returned, BALanCe-Partition calculates the



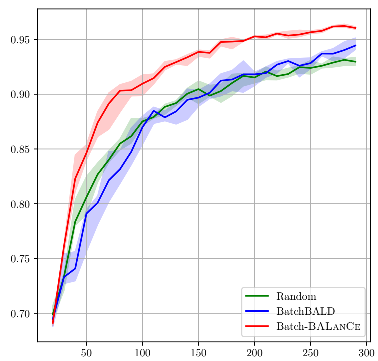
(a) repeat 0 time



(b) repeat 1 times



(c) repeat 2 times



(d) repeat 3 times

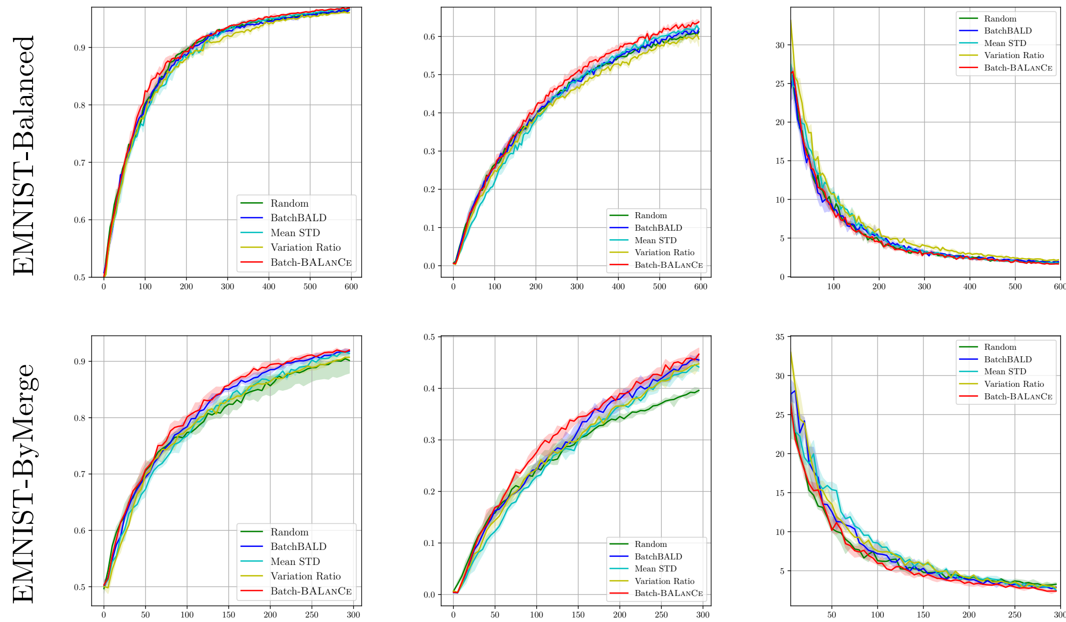
Figure 7: Performance of Random selection, BatchBALD, and Batch-BALanCe on Repeated-MNIST for an increasing number of repetitions. We can see that BatchBALD also performs worse as the number of repetitions is increased. Batch-BALanCe outperforms BatchBALD with large margins and remains similar performance across different numbers of repetitions.

edges discounts of all edges that connect different equivalence classes and estimates the acquisition function values of each data sample in the AL pool.

Although a faster method that utilizes complete homogeneous symmetric polynomials (Javdani et al., 2014) can be implemented to estimate the acquisition function values for BALanCe-Partition, experiments in Fig. 9 show that BALanCe-Partition can not achieve better performance than BALanCe and increasing the MC dropout number does not improve performance significantly.

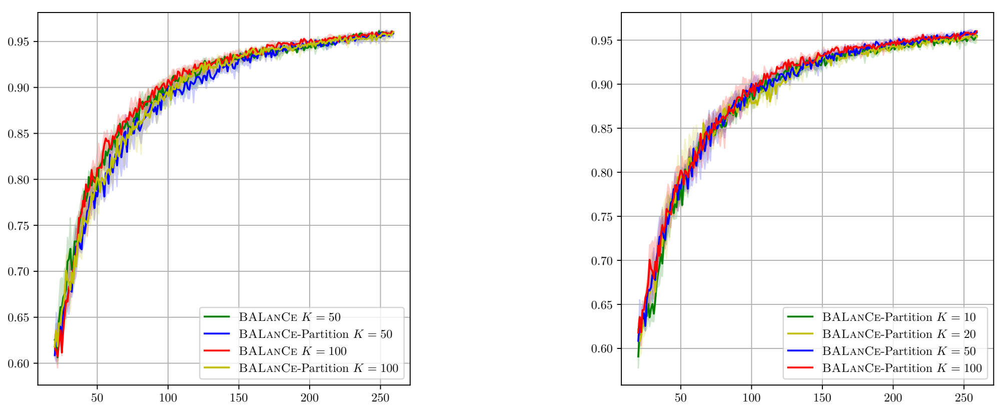
F.4 Coefficient of variation

To gain more insight into why BALanCe and Batch-BALanCe work consistently better than BALD and BatchBALD, we further investigate the dispersion of the estimated acquisition function values for those methods. Since Batch-BALanCe and BatchBALD



(a) Macro-average AUC (b) Macro-average F1 (c) NLL

Figure 8: Compare different metrics for EMNIST-Balanced and EMNIST-Bymerge



(a) Compare BALanCe-Partition with BALanCe (b) BALanCe-Partition with different K

Figure 9: Accuracy vs. # queries for BALanCe-Partition and BALanCe.

extend their fully sequential algorithms similarly in a greedy manner, we only compare the acquisition functions of BALanCe and BALD.

The coefficient of variation (CV) is chosen for the comparison of dispersion. It is defined as the ratio of the standard deviation to the mean. CV is a standardized measure of the dispersion of a probability distribution or frequency distribution. The value of CV is independent of the unit in which it is taken.

Method	repeat 0 time	repeat 1 times	repeat 2 times	repeat 3 times
Random	0.887 ± 0.017	0.883 ± 0.012	0.881 ± 0.013	0.895 ± 0.009
BatchBALD	0.917 ± 0.005	0.892 ± 0.023	0.883 ± 0.025	0.881 ± 0.014
Batch-BALanCe	0.926 ± 0.008	0.923 ± 0.008	0.929 ± 0.004	0.927 ± 0.010

Table 3: Mean±STD of test accuracies when acquired training set size is 130

We conduct the experiment on the imbalanced MNIST dataset in §A.2. We estimate the acquisition function values of BALanCe and BALD 5 times with 5 sets of K MC dropouts for each sample in the AL pool. Then, the CVs are calculated for these estimations. In Fig. 10, we show histograms of CVs for both methods. The estimated acquisition function values of BALanCe are less dispersed, which shows potential for better performance.

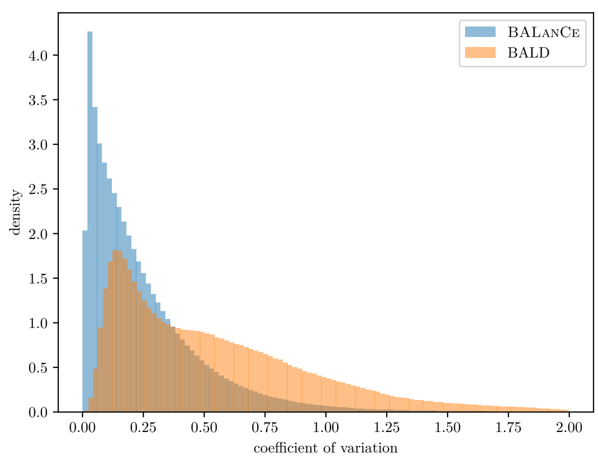


Figure 10: Histograms for coefficient of variation.