SABAL: Sparse Approximation-based Batch Active Learning

Maohao Shen*
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
maohaos2@illinois.edu

Bowen Jiang*
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
bowenj2@illinois.edu

Jacky Y. Zhang*
Department of Computer Science
University of Illinois at Urbana-Champaign
yiboz@illinois.edu

Oluwasanmi Koyejo
Department of Computer Science
University of Illinois at Urbana-Champaign
sanmi@illinois.edu

Abstract

We propose a novel and general framework (*i.e.*, SABAL) that formulates batch active learning as a sparse approximation problem. SABAL aims to find a weighted subset from the unlabeled data pool such that the corresponding training loss function approximates its full data pool counterpart. We realize the general framework as a sparsity-constrained discontinuous optimization problem that explicitly balances uncertainty and representation for large-scale applications, for which we propose both greedy and iterative hard thresholding schemes. The proposed method can adapt to various settings, including both Bayesian and non-Bayesian neural networks. Numerical experiments show that that SABAL achieves state-of-the-art performance across different settings with lower computational complexity.

1 Introduction

Over the last decade, deep neural networks have achieved promising results in various challenging supervised learning tasks, leveraged by the increases in computational power and availability of massive training datasets. However, obtaining labels for a complex training dataset can be challenging in practice, as the data annotation is usually a time-consuming process that may require professional knowledge in certain applications such as medical science [1, 2]. To mitigate the problem of scarce labeled data, *Active Learning* (AL) [3] is commonly employed as a powerful technique for efficient model training with limited annotation costs. Given a partially labeled dataset, active learning ideally selects which data samples are the best for learning. Specifically, it aims to iteratively query the most helpful data to ask an oracle (human annotator) to annotate. The queried data samples can be added back to the labeled data pool, and the model is updated. This process is repeated until the model has achieved a desired performance.

Intelligently identifying the most valuable data for annotation, also known as the query strategy, is the key problem in active learning. The common strategies usually take the prediction uncertainty or data representation as the metric for data selection. The *uncertainty-based* approach is the most prevalent query strategy [3, 4, 5, 6], which works by querying samples with low model prediction confidence or with high uncertainty. However, the uncertainty-based approach often results in selecting correlated and redundant data samples in each queried batch, which may result in worse model performance [7, 8]. Another common strategy is the *representation-based* approach [9, 10], which aims to select

*equal contribution

Preprint. Under review.

a subset of data that represents the whole unlabeled dataset. Nevertheless, representation-based active learning methods exhibit some weakness, since they tend to be computationally expensive and sensitive to batch sizes [11, 12]. More recently, several *hybrid* approaches that try to take both uncertainty and representation into consideration have shown advantages and are promising directions [11, 12, 13]. This paper takes this hybrid view towards an active learning framework that *balances the trade-off* between uncertainty and representation.

Besides the hybrid approaches, deep Bayesian active learning has also gained much attention due to recent advances in Bayesian deep learning. Several Bayesian approaches [5, 7] leverage the model uncertainty measurements [14, 15] determined by Bayesian neural networks, while the other recent works [16] leverage progress in Bayesian Coreset problems [17, 18, 19]. However, existing Bayesian approaches are explicitly designed for Bayesian Neural Networks. For example, the main procedure for approximating the log-posterior in [16] is not easily applied to non-Bayesian models. While we take some inspiration from the Bayesian paradigm, another goal of this paper is to propose a *general and flexible* method that is not limited to Bayesian models.

For modern deep models, it is reasonable to query a large batch of data simultaneously to reduce the model update frequency. This batch selection approach is known as *batch active learning*. Taking an optimization perspective, finding the best batch is NP-hard. Two common approaches for such combinatorial problems are: (1) greedy algorithms that select one data sample in sequence until the batch budget is exhausted [7, 20, 21]. Here, specific conditions of the acquisition function such as submodularity [22] are required to guarantee a good optimization result; (2) clustering algorithms regard cluster centers as their queried batch [9, 11], but the process is computationally expensive. To our knowledge, except for Pinsler et al. [16] that focus on the Bayesian models, so far active learning has rarely been studied from the perspective of *sparse approximation*. This is despite the ubiquity of sparse approximation in the signal processing community for tasks such as dictionary learning [23] and compressed sensing [24], due to its performance for discovering a sparse representation while avoiding redundant information. Here we employ sparse approximation methods for batch active learning tasks.

Our main contributions are summarized in the following. We propose a novel and flexible Sparse Approximation-based Batch Active Learning framework, *i.e.*, SABAL. We show how SABAL generalizes batch active learning as a sparse approximation problem and can adapt to different settings and models. The central intuition of SABAL is finding a *weighted subset* from the unlabeled data pool so that its corresponding training loss approximates the *full-set* loss function. We realize the SABAL framework as an efficient finite-dimensional optimization problem: First, we derive an upper bound to balance the trade-off between uncertainty and representation in a principled way. Second, we approximate the loss functions using finite-dimensional approximation. This results in a sparsity-constrained discontinuous optimization problem, for which we propose several efficient optimization algorithms. We demonstrate the advantages of SABAL in experiments for both Bayesian and general batch active learning settings.

The structure of this manuscript is as follows. In Section 2, we formulate the general framework of SABAL, and in Section 3, we realize the framework into a finite-dimensional discontinuous sparse optimization problem. To solve the resulting optimization problem, we propose two optimization algorithms in Section 4. Then, the results of our experiments are presented in section 5. Due to limited space, additional related work are discussed in Appendix Section A, and all proofs are provided in Appendix Section B.

2 Batch Active Learning as Sparse Approximation

This section introduces the preliminaries and the general formulation of batch active learning as a sparse approximation problem.

Preliminaries Vectors are denoted as bold lower case letters, *e.g.*, $\boldsymbol{w} \in \mathbb{R}^n$. The l_0 pseudo-norm of a vector $\boldsymbol{w}$ is denoted as $\|\boldsymbol{w}\|_0$, *i.e.*, the number of non-zero elements of $\boldsymbol{w}$. We denote $\mathbb{R}_+ := [0, +\infty)$. Distributions are denoted in script, *e.g.*, $\mathscr{P}$, and a random variable is denoted by tilde, *e.g.*, $\tilde{\boldsymbol{y}} \sim \mathscr{P}$. We denote sets in calligraphy or in uppercase Greek alphabet (*e.g.*, $\mathcal{D}, \Theta$), and additionally we denote $[n] := \{1, 2, \ldots, n\}$. In supervised learning, given a labeled training dataset $\mathcal{D}_l := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n_l}$, where we denote their domain to be $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$, the empirical goal is to

minimize a loss function $L_l(\boldsymbol{\theta}) := \sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\mathcal{D}_l} \ell(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\theta})$ formed by the training dataset, where $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^m$ is the parameter of the model and ℓ is a loss function evaluated on individual pairs of data. Without loss of generality, we assume $\Theta \subset \mathbb{R}^m$ is compact and $\ell(\boldsymbol{x}, \boldsymbol{y}; \cdot) : \Theta \to \mathbb{R}$ is in a normed space $(\mathcal{L}(\Theta, \mathbb{R}), \|\cdot\|_\dagger)$ for all $\boldsymbol{x}, \boldsymbol{y}$. We further assume the constant function $f : \Theta \to 1$ is included in $\mathcal{L}(\Theta, \mathbb{R})$. The "†" in the norm $\|\cdot\|_\dagger : \mathcal{L}(\Theta, \mathbb{R}) \to \mathbb{R}_+$, representing its definition is a placeholder that will be discussed later.

Batch Active Learning Besides the labeled dataset $\mathcal{D}_l$, there is an unlabeled dataset $\mathcal{D}_u := \{\boldsymbol{x}_j\}_{j=1}^{n_u}$ where the corresponding labels are not known but could be acquired at a high cost, *e.g.*, through human labeling. Combining the two datasets, denoting $n := n_l + n_u$, the ideal loss function to minimize would be

$$\sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\mathcal{D}_l} \ell(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\theta}) + \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} \ell(\boldsymbol{x}_j, \boldsymbol{y}_j^\star; \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{y}_j^\star$ is the unknown true label corresponding to the data $\boldsymbol{x}_j$. Since the true labels can only be acquired at a high cost, we have to impose a budget b ($b < n_u$) on the number of label acquisitions. Therefore, the *batch active learning* problem is to find a subset $\mathcal{S} \subset \mathcal{D}_u$ such that we can obtain a good model by optimizing the following loss function, *i.e.*,

$$\sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\mathcal{D}_l} \ell(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\theta}) + \sum_{\boldsymbol{x}_j\in\mathcal{S}} \ell(\boldsymbol{x}_j, \boldsymbol{y}_j^\star; \boldsymbol{\theta}), \quad \text{where } |\mathcal{S}| = b. \tag{2}$$

Generalized Batch Active Learning We start our method by generalize the classical formulation (equation 2) by considering an importance weight for each unlabeled data. That is, we aim to find a sparse non-negative vector $\boldsymbol{w} \in \mathbb{R}_+^{n_u}$ such that we can obtain a good model by optimizing the following loss function, *i.e.*,

$$\sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\mathcal{D}_l} \ell(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\theta}) + \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} w_j \ell(\boldsymbol{x}_j, \boldsymbol{y}_j^\star; \boldsymbol{\theta}), \quad \text{where } \|\boldsymbol{w}\|_0 = b. \tag{3}$$

A *key question* now is—what is the criterion for a good $\boldsymbol{w}$? Comparing the ideal loss function (equation 1) and the sparse importance weighted loss (equation 3), we note that the only difference is their unlabeled data loss functions. Therefore, a straight-forward informal criterion for a good importance weight $\boldsymbol{w}$ is that $\boldsymbol{w}$ makes the two unlabeled data loss functions close to each other, *i.e.*,

$$L_{\boldsymbol{w}}^\star(\boldsymbol{\theta}) := \frac{1}{b} \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} w_j \ell(\boldsymbol{x}_j, \boldsymbol{y}_j^\star; \boldsymbol{\theta}) \quad \approx \quad L^\star(\boldsymbol{\theta}) := \frac{1}{n_u} \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} \ell(\boldsymbol{x}_j, \boldsymbol{y}_j^\star; \boldsymbol{\theta}).$$

Note that since $\|\boldsymbol{w}\|_0 = b < n_u$, the average uses the number of non-zero components to account for the difference in the number of data samples. However, as the true labels are unknown, we cannot compute $L_{\boldsymbol{w}}^\star$ and $L^\star$. Luckily, we indeed can have an estimation for the true labels, *i.e.*, the estimation based on the labeled data $p(\tilde{\boldsymbol{y}}_j \mid \boldsymbol{x}_j, \mathcal{D}_l)$ or its various approximations. Denote $\mathscr{P}(\boldsymbol{x}_j)$ as an estimation distribution of the corresponding label $\boldsymbol{y}_j$, and the informal criterion for a good importance weight $\boldsymbol{w}$ then becomes

$$\tilde{L}_{\boldsymbol{w}}(\boldsymbol{\theta}) := \frac{1}{b} \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} w_j \ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta}) \quad \approx \quad \tilde{L}(\boldsymbol{\theta}) := \frac{1}{n_u} \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} \ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta}), \tag{4}$$

where $\tilde{\boldsymbol{y}}_j \sim \mathscr{P}(\boldsymbol{x}_j)$. Therefore, we are one step closer to be able to evaluate how optimal a weighted selection is, and the next question is how to measure the difference between $\tilde{L}$ and $\tilde{L}_{\boldsymbol{w}}$.

Difference Between Two Loss Functions Given the two loss functions $\tilde{L}, \tilde{L}_{\boldsymbol{w}} \in \mathcal{L}(\Theta, \mathbb{R})$, where $\mathcal{L}(\Theta, \mathbb{R})$ is equipped with the norm $\|\cdot\|_\dagger$, a straight-forward measurement of the difference between them is $\|\tilde{L} - \tilde{L}_{\boldsymbol{w}}\|_\dagger$. However, observing that the optimization of a loss function is shift-invariant, the difference between two loss functions should also be shift-invariant. For example, for $\forall L \in \mathcal{L}(\Theta, \mathbb{R})$ we have $\arg\min_{\boldsymbol{\theta}\in\Theta}(L(\boldsymbol{\theta}) + c) = \arg\min_{\boldsymbol{\theta}\in\Theta} L(\boldsymbol{\theta})$ for $\forall c \in \mathbb{R}$, implying that $L + c$ should be treated the same as L. Therefore, to account for the shift-invariance, we define $q : \mathcal{L}(\Theta, \mathbb{R}) \to \mathbb{R}_+$ as

$$q(L) := \inf_{c\in\mathbb{R}} \|L + c\|_\dagger, \qquad \forall L \in \mathcal{L}(\Theta, \mathbb{R}). \tag{5}$$

Note that *we abuse the notation* a bit, *i.e.*, the c in $L + c$ should be the constant function that maps every $\boldsymbol{\theta} \in \Theta$ to c. The above definition has some nice properties that make it a good difference measurement of two loss functions, as shown in the following proposition.

Proposition 2.1. *$q : \mathcal{L}(\Theta, \mathbb{R}) \to \mathbb{R}_+$ defined in (5) is a shift-invariant seminorm satisfying the following properties. (1)* ***triangle inequality****: $q(L_1 + L_2) \leq q(L_1) + q(L_2)$ for $\forall L_1, L_2 \in \mathcal{L}(\Theta, \mathbb{R})$; (2)* ***absolute homogeneity****: $q(cL) = |c|q(L)$ for $\forall L \in \mathcal{L}(\Theta, \mathbb{R}), \forall c \in \mathbb{R}$; (3)* ***shift-invariance****: $q(L + c) = q(L)$ for $\forall L \in \mathcal{L}(\Theta, \mathbb{R}), \forall c \in \mathbb{R}$; (4) $q(L) = 0$ if and only if L maps every $\boldsymbol{\theta} \in \Theta$ to a constant. In other words, q defines a norm in the space of shift-equivalence classes of loss functions.*

Therefore, we can formulate the generalized batch active learning problem as the following sparse approximation problem.

Problem 1 (Sparse Approximation-based Batch Active Learning)**.** *Given the shift-invariant seminorm q induced by the norm $\|\cdot\|_\dagger$ (equation 5), and a label estimation distribution $\mathscr{P}$, the generalized batch active learning problem (equation 4) is formally defined as*

$$\arg\min_{\boldsymbol{w} \in \mathbb{R}_+^{n_u}} \quad \mathbb{E}_{\mathscr{P}}[q(\tilde{L} - \tilde{L}_{\boldsymbol{w}})] \qquad \textit{s.t.} \qquad \|\boldsymbol{w}\|_0 = b, \tag{6}$$

where $\mathbb{E}_{\mathscr{P}}$ stands for the expectation over $\tilde{\boldsymbol{y}}_j \sim \mathscr{P}(\boldsymbol{x}_j)$ for $\forall j \in [n_u]$.

Problem 1 (SABAL) offers a general framework for batch active learning and can be applied with various settings, *i.e.*, the norm $\|\cdot\|_\dagger$ and the individual loss function ℓ can be chosen based on use. In the next section, we introduce a practical realization of equation 6.

3 Sparse Approximation as Finite-dimensional Optimization

In this section, we transform the sparse approximation problem (equation 6) into a finite-dimensional sparse optimization problem. First, we address an issue regarding the sampling of $\mathbb{E}_{\mathscr{P}}$. Then, we discuss some concrete choices of $\mathscr{P}$ and $\|\cdot\|_\dagger$ that lead to a finite-dimensional sparse optimization.

Addressing the Sampling Issue In equation 6, the expectation $\mathbb{E}_{\mathscr{P}}$ is taken over the product space of $(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_{n_u})$ and each sample has to be remembered for future optimization, which can be intractable for large datasets. However, it has an upper bound where the complexity of the optimization is independent of the number of samples from $\mathscr{P}$. First, by the triangle inequality

$$\begin{aligned}\mathbb{E}_{\mathscr{P}}[q(\tilde{L} - \tilde{L}_{\boldsymbol{w}})] &= \mathbb{E}_{\mathscr{P}}[q(\tilde{L} - \mathbb{E}_{\mathscr{P}}[\tilde{L}] + \mathbb{E}_{\mathscr{P}}[\tilde{L}] - \mathbb{E}_{\mathscr{P}}[\tilde{L}_{\boldsymbol{w}}] + \mathbb{E}_{\mathscr{P}}[\tilde{L}_{\boldsymbol{w}}] - \tilde{L}_{\boldsymbol{w}})] \\ &\leq \underbrace{\mathbb{E}_{\mathscr{P}}[q(\tilde{L} - \mathbb{E}_{\mathscr{P}}[\tilde{L}])] + \mathbb{E}_{\mathscr{P}}[q(\tilde{L}_{\boldsymbol{w}} - \mathbb{E}_{\mathscr{P}}[\tilde{L}_{\boldsymbol{w}}])]}_{(i)\text{: variance}} + \underbrace{q(\mathbb{E}_{\mathscr{P}}[\tilde{L}] - \mathbb{E}_{\mathscr{P}}[\tilde{L}_{\boldsymbol{w}}])}_{(ii)\text{: approximation bias}}. \end{aligned} \tag{7}$$

We can see that it offers a trade-off between bias and variance, where the bias term is immediately tractable by expanding $\tilde{L}, \tilde{L}_{\boldsymbol{w}}$:

$$\begin{aligned}(ii) &= q(\mathbb{E}_{\mathscr{P}}[\tfrac{1}{n_u} \textstyle\sum_{\boldsymbol{x}_j \in \mathcal{D}_u} \ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta})] - \mathbb{E}_{\mathscr{P}}[\tfrac{1}{b} \textstyle\sum_{\boldsymbol{x}_j \in \mathcal{D}_u} w_j \ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta})]) \\ &= q((\tfrac{1}{n_u} \textstyle\sum_{\boldsymbol{x}_j \in \mathcal{D}_u} \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta})]) - (\tfrac{1}{b} \textstyle\sum_{\boldsymbol{x}_j \in \mathcal{D}_u} w_j \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta})])). \end{aligned} \tag{8}$$

It remains to address the variance term *(i)*. Recall that the more accurate $\mathscr{P}$ is, the more accurate our approximation is. Fortunately, we can obtain a free improvement of $\mathscr{P}$ given the decision of $\boldsymbol{w}$. That is, if we decide to acquire the label of $\boldsymbol{x}_j$ (*i.e.*, $w_j > 0$), we then know that the distribution of $\tilde{\boldsymbol{y}}_j$ given $\boldsymbol{x}_j$ will be concentrated on its true label $\boldsymbol{y}_j^\star$, *i.e.*,

$$\tilde{\boldsymbol{y}}_j \sim \mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j) := \begin{cases} \mathscr{P}(\boldsymbol{x}_j) & \text{if } w_j = 0 \\ \delta_{\boldsymbol{y}_j^\star} & \text{if } w_j > 0 \end{cases}, \qquad \text{where } \boldsymbol{w} \in \mathbb{R}_+^{n_u} \tag{9}$$

where $\delta_{\boldsymbol{y}_j^\star}$ denotes the distribution that $\tilde{\boldsymbol{y}}_j$ can only be $\boldsymbol{y}_j^\star$. The reason that this improvement can be applied to *(i)* but not *(ii)* is: given a decision $\boldsymbol{w}$, the true labels $\boldsymbol{y}_j^\star$ for $w_j > 0$ is still unknown, but it is known that the corresponding variance is zero.

Proposition 3.1. *As $\boldsymbol{w} \in \mathbb{R}_+^{n_u}$ and $\|\boldsymbol{w}\|_0 = b$, by replacing the $\mathscr{P}$ by the improved estimation distribution $\mathscr{P}_{\boldsymbol{w}}$ (equation 9) into (i) in equation 7, we have*

$$\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L} - \mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}])] + \mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L}_{\boldsymbol{w}} - \mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}_{\boldsymbol{w}}])] \leq \sum_{\boldsymbol{x}_j \in \mathcal{D}_u} \mathbf{1}(w_j = 0) \cdot \sigma_j, \tag{10}$$

where $\sigma_j := \frac{1}{n_u} \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[q(\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \cdot) - \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \cdot)])]$ is the individual variance, and $\mathbf{1}(\cdot)$ is the indicator function.

Therefore, combining equation 8 and equation 10, we have a more tractable form of the sparse approximation, *i.e.*,

$$\underset{\boldsymbol{w}\in\mathbb{R}_+^{n_u}}{\arg\min} \quad q(\mathbb{E}_{\mathscr{P}}[\tilde{L}] - \mathbb{E}_{\mathscr{P}}[\tilde{L}_{\boldsymbol{w}}]) + \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} \mathbf{1}(w_j = 0)\cdot\sigma_j \qquad \text{s.t.} \qquad \|\boldsymbol{w}\|_0 = b, \tag{11}$$

and it remains to specify the choice of $\|\cdot\|_\dagger$, *i.e.*, the norm that induces q (equation 5).

Formulation of the Finite-Dimensional Optimization The distribution $\mathscr{P}(\boldsymbol{x}_j) = p(\tilde{y}_j \mid \boldsymbol{x}_j, \mathcal{D}_l)$ can be directly applied on Bayesian models. For non-Bayesian models, one could utilize the MC Dropout [5] to approximate the predictive posterior. We make the norm $\|\cdot\|_\dagger$ more concrete by considering a sampling distribution π of $\boldsymbol{\theta}$, *i.e.*, for $\forall L \in \mathcal{L}(\Theta, \mathbb{R}) : \|L\|_\pi^2 = \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[L(\boldsymbol{\theta})^2]$. Accordingly,

$$q(L)^2 = \inf_{c\in\mathbb{R}} \|L + c\|_\pi^2 = \inf_{c\in\mathbb{R}} \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[(L(\boldsymbol{\theta}) + c)^2] = \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[(L(\boldsymbol{\theta}) - \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[L(\boldsymbol{\theta})])^2]. \tag{12}$$

The distribution π tells us where and how to evaluate the "magnitude" of L, and we discuss two ways to approximate equation 12.

1. When it is easy to sample from the posterior $p(\boldsymbol{\theta} \mid \mathcal{D}_l)$, we can draw m samples $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta} \mid \mathcal{D}_l)$. Denote $\boldsymbol{g} := \frac{1}{\sqrt{m}}[\ldots, (L(\boldsymbol{\theta}_i) - \bar{L}), \ldots]_{i=1\ldots m}^\top \in \mathbb{R}^m$ where $\bar{L} := \frac{1}{m}\sum_{i=1}^m L(\boldsymbol{\theta}_i)$. From equation 12 we have
$$q(L)^2 \approx \frac{1}{m}\sum_{i=1}^m (L(\boldsymbol{\theta}_i) - \bar{L})^2 = \|\boldsymbol{g}\|_2^2, \tag{13}$$
where $\|\boldsymbol{g}\|_2$ is simply the Euclidean norm of the m-dimensional vector $\boldsymbol{g}$.
2. When Θ (the space of $\boldsymbol{\theta}$) is small, or the loss functions L is smooth, we can approximate it by $L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}_0) + \nabla L(\boldsymbol{\theta}_0)^\top(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$, where $\boldsymbol{\theta}_0$ can be the parameter of the current model. From equation 12 we have
$$\begin{aligned} q(L)^2 &\approx \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[(\nabla L(\boldsymbol{\theta}_0)^\top\boldsymbol{\theta} - \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[\nabla L(\boldsymbol{\theta}_0)^\top\boldsymbol{\theta}])^2] = \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[(\nabla L(\boldsymbol{\theta}_0)^\top(\boldsymbol{\theta} - \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[\boldsymbol{\theta}])^2] \\ &\le \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[\|\boldsymbol{\theta} - \mathbb{E}_{\boldsymbol{\theta}\sim\pi}[\boldsymbol{\theta}]\|_2^2]\cdot\|\nabla L(\boldsymbol{\theta}_0)\|_2^2, \end{aligned} \tag{14}$$
which is the result of applying the Cauchy-Schwarz inequality. Note that $\|\nabla L(\boldsymbol{\theta}_0)\|_2$ is the Euclidean norm of the gradient vector $\nabla L(\boldsymbol{\theta}_0) \in \mathbb{R}^m$.

Plugging either of the two approximations of $q(L)$ into equation 11, and squaring all of the terms for the ease of optimization, we can formulate the sparse approximation problem as the following finite-dimensional optimization problem, where $\alpha > 0$ offers a trade-off between bias and variance.

$$\underset{\boldsymbol{w}\in\mathbb{R}_+^{n_u}}{\arg\min} \quad \|\boldsymbol{v} - \Phi\boldsymbol{w}\|_2^2 + \alpha\sum_{\boldsymbol{x}_j\in\mathcal{D}_u} \mathbf{1}(w_j = 0)\cdot\sigma_j^2 \quad \text{s.t.} \quad \|\boldsymbol{w}\|_0 = b, \tag{15}$$

where we denote $\boldsymbol{v} \in \mathbb{R}^m$, $\Phi \in \mathbb{R}^{m\times n_u}$ and σ_j as

$$\boldsymbol{v} := \frac{1}{n_u}\sum_{j=1}^{n_u} \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j)], \qquad \Phi := \frac{1}{b}(\mathbb{E}_{\mathscr{P}(\boldsymbol{x}_1)}[\boldsymbol{g}_1(\tilde{\boldsymbol{y}}_1)], \ldots, \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_{n_u})}[\boldsymbol{g}_{n_u}(\tilde{\boldsymbol{y}}_{n_u})]), \tag{16}$$

$$\sigma_j = \frac{1}{n_u}\mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\|\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j) - \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j)]\|_2], \tag{17}$$

$$\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j) := \begin{cases} [\ldots, (\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta}_i) - \bar{\ell}), \ldots]_{i=1\ldots m}^\top, \quad \bar{\ell} := \frac{1}{m}\sum_{i=1}^m \ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta}_i) & \text{if use (13)} \\ \nabla\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta}_0) & \text{if use (14).} \end{cases} \tag{18}$$

In practice, it is often the case that the number of parameters is less than the number of samples, *i.e.*, $m < n_u$. For over-parameterized neural networks, it is a common practice to use the gradient of the last layer to represent the full-model gradient [11, 25], and thus we usually still have $m < n_u$. Therefore, if we opt for a big label acquisition batch size $b > m$, the approximation bias $\|\boldsymbol{v} - \Phi\boldsymbol{w}\|_2^2$ then becomes under-determined, and thus there are possibly infinitely many $\boldsymbol{w}$ to make $\boldsymbol{v} = \Phi\boldsymbol{w}$, which means the optimization (equation 15) can be "overfitted". To make our method more stable in such settings, we include a ℓ_2 regularizer $\beta\|\boldsymbol{w} - \mathbf{1}\|_2^2$, where $\beta > 0$. Finally, noting that $w_j \ge 0$, we can see minimizing $\alpha\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\mathbf{1}(w_j = 0)\cdot\sigma_j^2$ is equivalent to minimizing $-\alpha\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\mathbf{1}(w_j > 0)\cdot\sigma_j^2$. Combining the above modifications, we have the following optimization problem.

Problem 2 (Sparse Approximation as Finite-dimensional Optimization)**.** *The finite-dimensional optimization for generalized batch active learning is*

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}_+^{n_u}} \quad \|\boldsymbol{v}-\Phi\boldsymbol{w}\|_2^2-\alpha\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\mathbf{1}(w_j>0)\cdot\sigma_j^2+\beta\|\boldsymbol{w}-\mathbf{1}\|_2^2 \quad \textit{s.t.} \quad \|\boldsymbol{w}\|_0=b. \tag{19}$$

While simplified, the result is a sparse discontinuous optimization problem that is generally difficult to solve. In the next section, we propose two optimization algorithms for equation 19 by exploiting its unique properties. The overall procedure of SABAL in practice is presented in Algorithm 1.

Algorithm 1: SABAL: Sparse Approximation-based Batch Active Learning

Input: Unpretrained model with intial parameters $\boldsymbol{\theta}$, initial unlabeled pool $\mathcal{D}_u$, initial labeled pool $\mathcal{D}_l=\varnothing$, initial number of data b_0, query batch size b, number of iterations T.

1 Query a random batch $\mathcal{S}_0$ of b_0 data from $\mathcal{D}_l$, update $\mathcal{D}_u\leftarrow\mathcal{D}_u\backslash\mathcal{S}_0$ and $\mathcal{D}_l\leftarrow\mathcal{D}_l\cup\mathcal{S}_0$.
2 Train the model using $\mathcal{S}_0$.
3 **for** $t=1,2,\ldots,T$ **do**
4 For each data $\boldsymbol{x}_j\in\mathcal{D}_u$, estimate its label distribution $\tilde{\boldsymbol{y}}_j\sim\mathscr{P}(\boldsymbol{x}_j)$.
5 Compute vector $\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j)$ either by random projection or gradient embedding using 18.
6 Compute $\boldsymbol{v}$ and Φ for the approximation bias term in equation 16.
7 Compute the variance term σ_j using equation 17 for each data $\boldsymbol{x}_j\in\mathcal{D}_u$.
8 Combine the approximation bias term and variance term into objective in equation 19.
9 Find sparse weight $\boldsymbol{w}$ s.t. $\|\boldsymbol{w}\|_0=b$ by optimizing the objective specified in section 4.
10 Select a batch of data $\mathcal{S}_t=\{\boldsymbol{x}_j\in\mathcal{D}_u\mid\boldsymbol{w}_j>0\}$ and query their labels.
11 Update $\mathcal{D}_u\leftarrow\mathcal{D}_u\backslash\mathcal{S}_t$ and $\mathcal{D}_l\leftarrow\mathcal{D}_l\cup\mathcal{S}_t$.
12 Reinitialize and retrain the model using updated $\mathcal{D}_l$, update model parameters $\boldsymbol{\theta}$
13 **end**

Return: Final model parameters $\boldsymbol{\theta}$.

4 Optimization Algorithms

In this section, we focus on how to optimize Problem 2. Denote the objective function in (19) as $f(\boldsymbol{w}):=f_1(\boldsymbol{w})+f_2(\boldsymbol{w})$, where

$$f_1(\boldsymbol{w}):=\|\boldsymbol{v}-\Phi\boldsymbol{w}\|_2^2+\beta\|\boldsymbol{w}-\mathbf{1}\|_2^2, \qquad f_2(\boldsymbol{w}):=-\alpha\textstyle\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\mathbf{1}(w_j>0)\cdot\sigma_j^2.$$

The optimization has two major difficulties, *i.e.*, the nonconvex sparsity constraint $\|\boldsymbol{w}\|_0=b$ and the discontinuous objective function f_2. When it comes to sparsity-constrained optimization, there are two schemes that are widely considered — greedy (Frank-Wolfe [19]) and iterative hard thresholding (IHT) [17, 26]. However, Problem 2 introduces the new difficulty other than the sparsity constraint, *i.e.*, the discontinuous component f_2, rendering the regular IHT and greedy useless. Therefore, we design and propose two algorithms (Algorithm 2&3) specifically for Problem 2 under the two schemes respectively, while incorporating the discontinuity.

We introduce some notations used in this section. Given a vector $\boldsymbol{g}$, we denote $[\boldsymbol{g}]_+$ as $\boldsymbol{g}$ with its negative elements set to 0. For an index j, we denote g_j or $(\boldsymbol{g})_j$ to be its j^{th} element. For an index set $\mathcal{S}$, we denote $[\boldsymbol{g}]_{\mathcal{S}}$ to be the vector where $([\boldsymbol{g}]_{\mathcal{S}})_j=(\boldsymbol{g})_j$ if $j\in\mathcal{S}$ and $([\boldsymbol{g}]_{\mathcal{S}})_j=0$ if $j\notin\mathcal{S}$. Moreover, we denote $\boldsymbol{e}^j$ to be the unit vector where $(\boldsymbol{e}^j)_j=1$ and $(\boldsymbol{e}^j)_i=0$ for $\forall i\neq j$.

Although the two algorithms use different schemes, they share the same two sub-procedures, namely, a line search and de-bias step, which significantly improve the optimization performance of sparse optimization [17]. The line search sub-procedure (Algorithm 4 in appendix) optimally solves the problem $\arg\min_{\mu\in\mathbb{R}} f_1(\boldsymbol{w}-\mu\boldsymbol{u})$, *i.e.*, given a direction $\boldsymbol{u}$, what is the best step size μ to move the $\boldsymbol{w}$ along $\boldsymbol{u}$. The de-bias sub-procedure (Algorithm 5 in appendix) adjusts a sparse $\boldsymbol{w}$ in its own sparse support for a better solution.

Opt. Algorithm: Greedy The core idea of the greedy approach (Algorithm 2) is noted in line 3, where it chooses an index j to move a step of size τ that minimizes the objective, *i.e.*,

$$j\leftarrow\arg\min_{j\in[n_u]\backslash\mathcal{S}} \quad (f_1(\boldsymbol{w}+\tau\boldsymbol{e}^j)-f_1(\boldsymbol{w}))+(f_2(\boldsymbol{w}+\tau\boldsymbol{e}^j)-f_2(\boldsymbol{w})).$$

By approximating $f_1(\boldsymbol{w}+\tau\boldsymbol{e}^j)-f_1(\boldsymbol{w})$ by its first-order approximation $\langle\nabla f_1(\boldsymbol{w}),\tau\boldsymbol{e}^j\rangle$, and noting that $f_2(\boldsymbol{w}+\tau\boldsymbol{e}^j)-f_2(\boldsymbol{w})=-\alpha\sigma_j^2$, we have the greedy step (line 3) in Algorithm 2. After choosing the index j to include, line 5 chooses an optimal step to move, followed by a de-bias step that further improves the solution in the current sparse support $\text{supp}(\boldsymbol{w})$.

Opt. Algorithm: Proximal iterative hard thresholding The core idea of the proximal IHT (Algorithm 3) is noted in line 6, where it combines both the hard thresholding and the proximal operator. It minimizes the discontinuous f_2 in a neighbourhood of the solution $\boldsymbol{s}$ obtained by minimizing f_1, while satisfying the constraints. Noting that $\frac{1}{2}\|\boldsymbol{w}-\boldsymbol{s}\|_2^2+f_2(\boldsymbol{w})=\sum_{j\in[n_u]}(\frac{1}{2}(w_j-s_j)^2-\alpha\sigma_j^2)$, this step can be done optimally by simply picking the top-b elements, as shown in the following. Given a b-sparse support set $\mathcal{S}\subset[n_u]$, we can see that $\min_{\boldsymbol{w}\in\mathbb{R}_+^{n_u},\text{supp}(\boldsymbol{w})\subseteq\mathcal{S}}\sum_{j\in[n_u]}(\frac{1}{2}(w_j-s_j)^2-\alpha\sigma_j^2)=\sum_{j\in\mathcal{S}}(\frac{1}{2}[-s_j]_+^2-\alpha\sigma_j^2)$. Therefore, line 6 in Algorithm 3 can be done by: (1) find the b smallest $(\frac{1}{2}[-s_j]_+^2-\alpha\sigma_j^2)$, denoting the resulting b-sparse index set as $\mathcal{S}^\star$; (2) let $\boldsymbol{w}\leftarrow[[\boldsymbol{s}]_{\mathcal{S}^\star}]_+$. After this core step, a de-bias step improves the solution $\boldsymbol{w}$ within its sparse support, followed by a momentum step.

Algorithm 2: SABAL-Greedy

Parameter: sparsity b; step size τ.

1 $\boldsymbol{w}\leftarrow\boldsymbol{0}$; $\mathcal{S}\leftarrow\emptyset$
2 **repeat**
3 $\quad j\leftarrow\underset{j\in[n_u]\setminus\mathcal{S}}{\arg\min}\ \tau(\nabla f_1(\boldsymbol{w}))_j-\alpha\sigma_j^2$
4 $\quad\mathcal{S}\leftarrow\mathcal{S}\cup\{j\}$ *(update selection)*
5 $\quad\mu\leftarrow\text{LineSearch}(\boldsymbol{e}^j,\boldsymbol{w})$
6 $\quad\boldsymbol{w}\leftarrow\text{De-bias}(\boldsymbol{w}-\mu\boldsymbol{e}^j)$
7 $\quad w_j\leftarrow 0$ for $\forall w_j<0$ $(\boldsymbol{w}\in\mathbb{R}_+^{n_u})$
8 **until** $|\mathcal{S}|=b$;

Return: $\boldsymbol{w}$

Algorithm 3: SABAL-IHT

Parameter: sparsity b; number of iterations T.

1 $\boldsymbol{w}\leftarrow\boldsymbol{0}$; $\boldsymbol{z}\leftarrow\boldsymbol{0}$
2 **repeat**
3 $\quad\boldsymbol{w}'\leftarrow\boldsymbol{w}$ *(save previous $\boldsymbol{w}$)*
4 $\quad\mu\leftarrow\text{LineSearch}(\nabla f_1(\boldsymbol{z}),\boldsymbol{z})$
5 $\quad\boldsymbol{s}\leftarrow\boldsymbol{z}-\mu\nabla f_1(\boldsymbol{z})$ *(gradient descent)*
6 $\quad\boldsymbol{w}\leftarrow\underset{\boldsymbol{w}\in\mathbb{R}_+^{n_u},\|\boldsymbol{w}\|_0\le b}{\arg\min}\ \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{s}\|_2^2+f_2(\boldsymbol{w})$
7 $\quad\boldsymbol{w}\leftarrow\text{De-bias}(\boldsymbol{w})$
8 $\quad w_j\leftarrow 0$ for $\forall w_j<0$ $(\boldsymbol{w}\in\mathbb{R}_+^{n_u})$
9 $\quad\tau\leftarrow\text{LineSearch}(\boldsymbol{w}-\boldsymbol{w}',\boldsymbol{w})$
10 $\quad\boldsymbol{z}\leftarrow\boldsymbol{w}-\tau(\boldsymbol{w}-\boldsymbol{w}')$ *(momentum)*
11 **until** T *iterations*;

Return: $\boldsymbol{w}$

Complexity Analysis We analyze the time complexity of the proposed algorithms with respect to the input size, *i.e.*, the number of data samples n, and the batch size b of batch active learning. As we can see, except for the line 6 in Algorithm 3, all steps are of time complexity $O(n)$. The line 6 in Algorithm 3 is finding the b smallest elements, which can be done in $O(n\log(b))$. Therefore, the time complexity for SABAL-Greedy is $O(nb)$, and the time complexity for SABAL-IHT is $O(n\log(b))$. Comparing to the time complexity $O(nb^2)$ of the state-of-the-art method BADGE [11], the two proposed algorithms can be much faster, especially with a large batch size b used in practice.

5 Experiment results

We demonstrate that SABAL is a flexible batch active learning framework by evaluating its performance on image classification tasks with various models under different settings. First, using Bayesian neural networks, we show the effectiveness of SABAL on Bayesian batch active learning. Next, we demonstrate that SABAL can also adapt well to general batch active learning with general neural networks. Finally, we conduct an ablation study to show how SABAL is able to balance the trade-offs between uncertainty and representation.

We have a fixed training set, validation set, and testing set in each of our experiments. The model is initially trained on small amounts of labeled data randomly selected from the training set and then iteratively performs the data acquisition and data annotation. The model is reinitialized and retrained at the beginning of each active learning iteration. After the model is well trained, its testing accuracy is evaluated on the testing set as a measure of the performance. All experiments are repeated multiple times, and the results are reported as mean and standard deviations shown in learning curves. All large models use five different seeds, except for the small model (LeNet-5 [27]) uses three different seeds. We implement SABAL using both proximal IHT and greedy as two different optimization methods for the sparse approximation (denoted as *SABAL-IHT* and *SABAL-Greedy*), and compare the performance

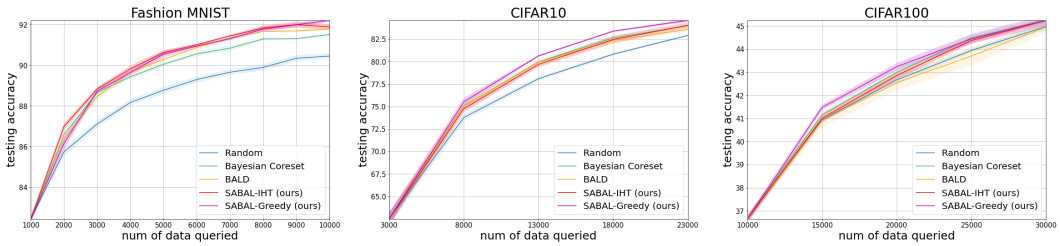

Figure 1: Active learning results on Bayesian models. Solid lines and shaded areas represent means and standard deviations of test accuracy respectively over different seeds.

with the following baselines from literature: **(1) Random**: A naive baseline that selects a batch of data uniformly at random. **(2) BALD** [28]: An uncertainty-based method that selects a batch of data with maximum mutual information between model parameters and predictions. Mutual information is defined as $\mathbb{I}\left(\boldsymbol{y}_i; \boldsymbol{\theta} \mid \boldsymbol{x}_i, \mathcal{D}_l\right) = \mathbb{H}\left(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \mathcal{D}_l\right) - \mathbb{E}_{\mathrm{p}(\boldsymbol{\theta}|\mathcal{D}_l)}\left[\mathbb{H}\left(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}, \mathcal{D}_l\right)\right]$. **(3) Entropy** [29]: An uncertainty-based method that selects a batch of data with maximum entropy of the model predictions $\mathbb{H}\left(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta}\right)$. **(4) KCenter** [9]: A representation-based method that reformulates the core-set selection as a k-Center problem in the feature embedding space. **(5) BADGE** (Batch Active Learning by Diverse Gradient Embeddings) [11]: A hybrid method that samples a diverse batch of data with high gradient magnitudes. It utilizes the k-MEANS++ seeding algorithm, which selects new centroids by iteratively sampling data points proportional to their distances to already selected centroids. It tries to select a batch that approximates the complete data log posterior, and formulate as a sparse approximation problem solved by Frank-Wolfe algorithm [30]. **(6) Bayesian Coreset** [16]: A Bayesian batch active learning approach based on Bayesian Coreset problem [18, 19].

SABAL for Bayesian Active Learning We first implement our experiments on Bayesian neural networks and perform Bayesian active learning on Fashion MNIST [31], CIFAR-10 [32], and CIFAR-100 [32]. We follow the experiment settings of [16], using a Bayesian neural network consisting of a ResNet-18 [33] feature extractor. The posterior inference is obtained by variational inference [34, 35] at the last layer. We estimate model predictive posteriors $p(\tilde{\boldsymbol{y}}_j \mid \boldsymbol{x}_j, \mathcal{D}_l)$ using 100 samples, and solve the finite-dimensional optimization problem with random projections as in equation 13. Under Bayesian settings, besides Random, we mainly focus on comparing with state-of-art approaches specifically designed for Bayesian active learning: BALD and Bayesian Coreset.

We then evaluate the performance of SABAL on Fashion MNIST dataset. We use 100 samples for random projections, 1000 seed data, and select 1000 data samples as the queried batch for 9 iterations. We then evaluate the performance of SABAL on CIFAR-10 and CIFAR-100, two more complicated datasets, using 2000 samples for random projections. We use 3000 (10000 for CIFAR-100) seed data and select 5000 data samples as the queried batch for 4 iterations. Because Bayesian Coreset usually finds a much smaller batch than requested, for a fair comparison, we let Bayesian Coreset acquire more data than the batch size, and stop the acquisition as long as it has selected a full batch of data.

It can be seen that both SABAL-IHT and SABAL-Greedy show some advantages on Fashion MNIST dataset. On CIFAR10 and CIFAR 100, we find SABAL-Greedy performs better than SABAL-IHT while outperforms other baselines, including Bayesian Coreset, the current state-of-the-art approach in the literature under Bayesian settings.

SABAL for General Active Learning We then implement our experiments on general convolutional neural networks, including LeNet-5 [27] and VGG-16 [36] architectures, using MNIST [31], SVHN [37], and CIFAR-10 [32] datasets. We utilize MC Dropout [5] to approximate the predictive posterior, as estimation of true label $p(\tilde{\boldsymbol{y}}_j \mid \boldsymbol{x}_j, \mathcal{D}_l)$, and solve the finite-dimensional optimization problem with gradient embedding, i.e., equation 14. With this general batch active learning setting, we mainly compare these baselines: Random, Entropy, BALD, KCenter, and BADGE.

We first evaluate the performance of SABAL on MNIST dataset with LeNet-5 model. We use 5 MC Dropout at inference time. We use 40 seed data and select 40 data samples as the queried batch for 15 iterations. We then evaluate the performance of SABAL on SVHN and CIFAR-10, which contains more complicated real-world color images, both of which are trained on the VGG-16 model. We use

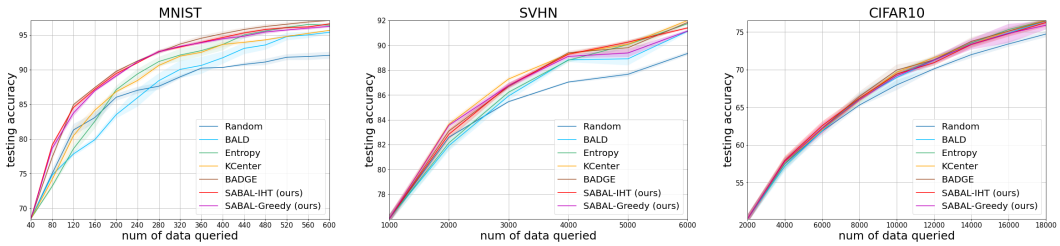

Figure 2: Active learning results on general (non-Bayesian) models. Solid lines and shaded areas represent means and standard deviations of test accuracy respectively over different seeds.

10 MC Dropout at inference time. We use 1000 (2000 for CIFAR10) seed data and select 1000 (2000) data samples as the queried batch for 5 (8) iterations.

Learning curves of test accuracy versus the number of data queried are shown in Figure 2. It turns out that both SABAL-IHT and SABAL-Greedy outperform most baselines, while achieving comparable performance to the strong baseline BADGE, the current state-of-the-art hybrid method in literature, but SABAL requires much less acquisition time compared with BADGE as stated in Section 4.

Ablation Study: trade-off of uncertainty and representation We perform an ablation study to understand better the trade-off between the variance and the bias terms in our final formulation equation 15. To remove the bias term, we query the data with top variances. To remove the contribution of variance, we query the data by only minimizing the approximation bias, i.e., setting $\alpha = 0$. We take two datasets MNIST and CIFAR-10 in the Bayesian experiment as examples. Results in Figure 3 demonstrate the necessity of taking both uncertainty and representation into consideration during the data acquisitions for ideal performance, while for some datasets like CIFAR-10, we discover variance's contribution is much more significant.

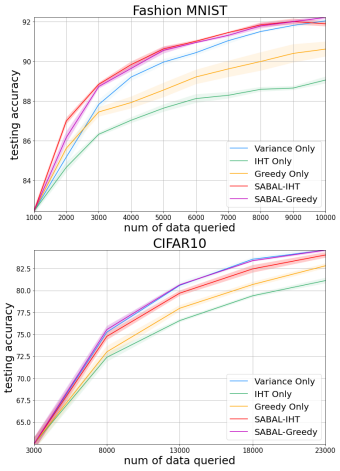

Figure 3: Ablation Study Results on Bayesian models.

Discussion We find that our method under Bayesian settings outperforms the baselines by a large margin, where one reason is that the Bayesian neural network itself is a strong tool to approximate posterior and sample model parameters. While on the general model, SABAL uses the gradient embedding as an upper bound approximation of the original loss function, and can only capture its local information. An alternative way is to use random projection by MC sampling, however, MC Dropout on general neural network suffers from its slow model inference. Besides, we find that SABAL-IHT and SABAL-Greedy have similar performances on general models. However, on Bayesian models with large-scale datasets, SABAL-IHT performances slightly worse than SABAL-Greedy. To figure out why, we notice that the main difference between SABAL under these two settings is the formulation of $\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j)$, which affects the following sparse approximation optimization step. On general neural networks, $\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j)$ is obtained by gradient embedding. On Bayesian neural networks, $\boldsymbol{g}_j(\tilde{\boldsymbol{y}}_j)$ is obtained by random projections, *i.e.*, sampling different model parameters from the posterior distribution. Since such sampling is usually taken around the neighborhood of current model parameter, *i.e.*, $\boldsymbol{\theta}_0$, the matrix $\Phi \in \mathbb{R}^{m \times n_u}$ would have nearly linearly dependent rows, which leads the approximation bias term $\|\boldsymbol{v} - \Phi \boldsymbol{w}\|_2^2$ to be ill-posed. Therefore, it's reasonable to believe that SABAL-Greedy is better than SABAL-IHT on Bayesian models because it is less sensitive to the "overfitting" problems during the optimization.

6 Conclusion

We introduce the SABAL as a novel framework that formulates batch active learning as a sparse approximation problem. It balances representation and uncertainty in a principled way, and has the

flexibility to adapt to various settings, *e.g.*, with Bayesian or non-Bayesian models. We realize the SABAL framework as a finite-dimensional optimization problem, efficiently solvable by the proposed greedy or proximal iterative hard thresholding algorithms. Numerical experiments demonstrate the strong performance of SABAL, comparable or better than the state-of-the-art and most baseline methods with lower complexities. For the future works, although the hyperparameter α offers a controllable trade-off between the variance and bias, it is still not well-understood how to strike the best balance. An in-depth theoretical analysis of the SABAL framework and the optimizations would also have the potential to inspire discoveries of even better batch active learning algorithms.

References

[1] Steven CH Hoi, Rong Jin, Jianke Zhu, and Michael R Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on Machine learning*, pages 417–424, 2006.

[2] Maohao Shen, Jacky Y. Zhang, Leihao Chen, Weiman Yan, Neel Jani, Brad Sutton, and Oluwasanmi Koyejo. Labeling cost sensitive batch active learning for brain tumor segmentation. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 1269–1273, 2021. doi: 10.1109/ISBI48211.2021.9434098.

[3] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[4] 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.

[5] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017.

[6] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018.

[7] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pages 7026–7037, 2019.

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

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

[10] Yazhou Yang and Marco Loog. Single shot active learning using pseudo annotators. *Pattern Recognition*, 89:22–31, 2019.

[11] 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.

[12] Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. Deep active learning: Unified and principled method for query and training. In *International Conference on Artificial Intelligence and Statistics*, pages 1308–1318. PMLR, 2020.

[13] 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.

[14] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Insights and applications. In *Deep Learning Workshop, ICML*, volume 1, page 2, 2015.

[15] 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.

[16] Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, pages 6359–6370, 2019.

[17] Jacky Y. Zhang, Rajiv Khanna, Anastasios Kyrillidis, and Oluwasanmi Koyejo. Bayesian coresets: Revisiting the nonconvex optimization perspective. In *International Conference on Artificial Intelligence and Statistics*, pages 2782–2790. PMLR, 2021.

[18] Jonathan H Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. *arXiv preprint arXiv:1605.06423*, 2016.

[19] Trevor Campbell and Tamara Broderick. Automated scalable bayesian inference via hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

[20] Erdem Bıyık, Kenneth Wang, Nima Anari, and Dorsa Sadigh. Batch active learning using determinantal point processes. *arXiv preprint arXiv:1906.07975*, 2019.

[21] Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. In *International Conference on Machine Learning*, pages 160–168. PMLR, 2013.

[22] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[23] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[24] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4): 1289–1306, 2006.

[25] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[26] Rajiv Khanna and Anastasios Kyrillidis. Iht dies hard: Provable accelerated iterative hard thresholding. In *International Conference on Artificial Intelligence and Statistics*, pages 188–198. PMLR, 2018.

[27] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5):14, 2015.

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

[29] Dan Wang and Yi Shang. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*, pages 112–119. IEEE, 2014.

[30] Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[31] 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.

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

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] Martin J Wainwright and Michael Irwin Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

[35] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[38] Sanjoy Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781, 2011.

[39] Steve Hanneke et al. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3):131–309, 2014.

[40] Patrick Hemmer, Niklas Kühl, and Jakob Schöffer. Deal: Deep evidential active learning for image classification. *arXiv preprint arXiv:2007.11344*, 2020.

[41] Xin Li and Yuhong Guo. Adaptive active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866, 2013.

[42] 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.

[43] Yonatan Geifman and Ran El-Yaniv. Deep active learning over the long tail. *arXiv preprint arXiv:1711.00941*, 2017.

[44] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] **See the discussion in Section 5.**
   (c) Did you discuss any potential negative societal impacts of your work? [No] **Since what we propose is a general active learning algorithm for more efficient model training and data acquisitions, we do not expect negative societal impacts of our work.**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] **See Section B in Appendix.**
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] **See our supplemental material.**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] **See Section 5 and D in Appendix.**

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] **Solid lines and shaded area in all of our figures represent means and standard deviations of test accuracy over different seeds.**

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] **See Section D in Appendix.**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [No] **The datasets used in this paper are all popular public datasets. According to paperswithcode.com/datasets, SVHN has license type of custom (non-commercial), Fashion MNIST has license type of MIT, while all others has unknown license.**

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] **We only used existing popular datasets, which provides no information about the consent.**

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] **Except for CIFAR-100 that contains only low-resolution human images, all other datasets we used don't contain human images, so they can't have any personally identifiable or offensive information.**

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

SABAL: Sparse Approximation-based Batch Active Learning

Appendix

A Related Work

Active learning has been widely studied by the machine learning community. As most classic approaches have already been discussed in a detail in [3, 38, 39], we will briefly review some recent works in deep active learning.

Existing query strategies can mainly be categorized as uncertainty-based and representation-based. Uncertainty-based approaches look for data samples the model is mostly uncertain about. Meanwhile, under the Bayesian setting, several recent works leverage the Bayesian neural network to well measure the model uncertainty. Gal and Ghahramani [14, 15] proves the Monte-Carlo dropout (MC Dropout) as an approximation of performing Bayesian inference, and enables efficient uncertainty estimations in neural networks. Gal et al. [5] utilizes MC Dropout for approximating posterior distributions and adapts [28] as their uncertainty based acquisition function, and similarly, Kirsch et al. [7] proposes a batch-mode approach based on [5] and shows some improvements through a more accurate measurement of mutual information between the data batch and model parameters. While MC Dropout becomes prevalent for uncertainty estimation, Beluch et al. [6] shows ensemble-based methods lead to better performance because of more calibrated uncertainty estimation, and another recent work [40] also proposes a new uncertainty estimation method by replacing the softmax output of a neural network with the parameter of Dirichlet density. Other non-Bayesian approaches sometimes combine uncertainty estimation with other metrics: Li and Guo [41] combines an information density measure to maximize the mutual information between selected samples and remaining unlabeled samples under the Gaussian Process setting. Wang et al. [42] selects data based on several classic uncertainty metrics and incorporate a cost-efficient strategy by pseudo labeling the confident samples.

Representation-based approaches attempt to query diverse data samples that could best represent the overall unlabeled dataset. A recent work proposed by Sener and Savarese [9] defines the active learning as a core-set selection problem. They derive an upper bound for the core-set loss and construct representative batches by solving a k-Center problem in the feature space. In [43], the authors also explore the deep active learning with core-sets, but build the core-sets in the farthest-first compression scheme.

Data acquisitions with the trade-off between uncertainty and representation attract some attentions recently. Ash et al. [11] capture the uncertainty through the lens of gradients, and sample diverse batches on the gradient embedding by the k-MEANS++ seeding algorithm. Shui et al. [12] minimize the loss by uncertainty while using the distribution matching with Wasserstein distances to encourage the diversity. Sinha et al. [13] train a Variational Autoencoder and a discriminator in an adversarial fashion. The discriminator predicts a sample as unlabeled based on its likelihood of representativeness, and a batch of samples with the lowest confidence will be queried.

B Proofs

Proposition B.1 (Proposition 2.1 Restated)**.** *$q : \mathcal{L}(\Theta, \mathbb{R}) \to \mathbb{R}_+$ defined in (5) is a shift-invariant seminorm satisfying the following properties:*

1. $q(L_1 + L_2) \leq q(L_1) + q(L_2)$ *for* $\forall L_1, L_2 \in \mathcal{L}(\Theta, \mathbb{R})$; *(triangle inequality)*
2. $q(cL) = |c| q(L)$ *for* $\forall L \in \mathcal{L}(\Theta, \mathbb{R}), \forall c \in \mathbb{R}$; *(absolute homogeneity)*
3. $q(L + c) = q(L)$ *for* $\forall L \in \mathcal{L}(\Theta, \mathbb{R}), \forall c \in \mathbb{R}$; *(shift-invariance)*
4. $q(L) = 0$ *if and only if L maps every $\boldsymbol{\theta} \in \Theta$ to a constant.*

In other words, q defines a norm in the space of shift-equivalence classes of loss functions.

Proof. Recall that

$$q(L) := \inf_{c\in\mathbb{R}} \|L + c\|_\dagger, \qquad \forall L \in \mathcal{L}(\Theta, \mathbb{R}).$$

We prove the four properties respectively in the following.

1. The triangle inequality is inherited from the sub-additivity of the norm $\|\cdot\|_\dagger$. For $\forall L \in \mathcal{L}(\Theta, \mathbb{R})$, we have
$$\begin{aligned} q(L_1 + L_2) &= \inf_{c\in\mathbb{R}} \|L_1 + L_2 + c\|_\dagger = \inf_{c_1,c_2\in\mathbb{R}} \|L_1 + L_2 + c_1 + c_2\|_\dagger \\ &\le \inf_{c_1,c_2\in\mathbb{R}} \|L_1 + c_1\|_\dagger + \|L_2 + c_2\|_\dagger \\ &= (\inf_{c\in\mathbb{R}} \|L_1 + c\|_\dagger) + (\inf_{c\in\mathbb{R}} \|L_2 + c\|_\dagger) \\ &= q(L_1) + q(L_2). \end{aligned}$$

2. The absolute homogeneity is also inherited from the absolute homogeneity of the norm $\|\cdot\|_\dagger$. The case for $c = 0$ is obvious, and for $c \neq 0$ we have
$$\begin{aligned} q(cL) = |c|q(L) &= \inf_{c_1\in\mathbb{R}} \|cL + c_1\|_\dagger = \inf_{c_1\in\mathbb{R}} |c| \cdot \|L + c_1/c\|_\dagger \\ &= \inf_{c_2\in\mathbb{R}} |c| \cdot \|L + c_2\|_\dagger = |c|q(L). \end{aligned}$$

3. By the definition of $q(\cdot)$, we have the shift-invariance of $q(\cdot)$.

4. The "if" part can be proved by definition, *i.e.*, $q(c) = \inf_{c_1\in\mathbb{R}} \|c_1 + c\|_\dagger = \|0\|_\dagger = 0$.

   For the "only if" part, we need to be more rigorous by defining f_c to be the function that maps Θ to $c \in \mathbb{R}$. We further define $\mathcal{F} := \{f_c \mid c \in \mathbb{R}\} \subset \mathcal{L}(\Theta, \mathbb{R})$, and we can see $(\mathcal{F}, \|\cdot\|_\dagger)$ is a one-dimensional normed space. Letting $L \in \mathcal{L}(\Theta, \mathbb{R})$ and $q(L) = 0$, we have
   $$\inf_{f_{c_\epsilon}\in\mathcal{F}} \|L + f_c\|_\dagger = 0.$$
   Therefore, for $\forall \epsilon > 0$, $\exists c_\epsilon \in \mathbb{R}$ such that
   $$\begin{aligned} &\|L + f_{c_\epsilon}\|_\dagger \le \epsilon \\ \implies &\|f_{c_\epsilon}\|_\dagger = \|L + f_{c_\epsilon} - L\|_\dagger \le \epsilon + \|L\|_\dagger. \end{aligned}$$
   That being said, for $0 < \epsilon < 1$, we have $\|f_{c_\epsilon}\|_\dagger \le 1 + \|L\|_\dagger$. Denote $\mathcal{B} = \{f_c \in \mathcal{F} \mid \|f_c\|_\dagger \le 1 + \|L\|_\dagger\}$, and we can see $\mathcal{B}$ is a closed ball in $\mathcal{F}$. As $\mathcal{F}$ is one-dimensional, by Riesz's lemma we have $\mathcal{B}$ compact.

   As $\lim_{\epsilon\to 0} \|L + f_{c_\epsilon}\|_\dagger = 0$, *i.e.*, $f_{c_\epsilon} \to L$, by the compactness of $\mathcal{B}$ we have $L \in \mathcal{B}$. Therefore, L is also a constant function. Note that this conclusion does not require $\mathcal{L}(\Theta, \mathbb{R})$ to be complete.

□

Proposition B.2 (Proposition 3.1 Restated)**.** *As $\boldsymbol{w} \in \mathbb{R}_+^{n_u}$ and $\|\boldsymbol{w}\|_0 = b$, by replacing the $\mathscr{P}$ by the improved estimation distribution $\mathscr{P}_{\boldsymbol{w}}$ (equation 9) into (i) in equation 7, we have*

$$\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L} - \mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}])] + \mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L}_{\boldsymbol{w}} - \mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}_{\boldsymbol{w}}])] \le \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} \mathbf{1}(w_j = 0) \cdot \sigma_j,$$

where $\sigma_j := \frac{1}{n_u}\mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[q(\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \cdot) - \mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \cdot)])]$ is the individual variance, and $\mathbf{1}(\cdot)$ is the indicator function.

Proof. Recall that

$$\tilde{L}_{\boldsymbol{w}}(\boldsymbol{\theta}) := \frac{1}{b} \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} w_j \ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta}), \qquad \tilde{L}(\boldsymbol{\theta}) := \frac{1}{n_u} \sum_{\boldsymbol{x}_j\in\mathcal{D}_u} \ell(\boldsymbol{x}_j, \tilde{\boldsymbol{y}}_j; \boldsymbol{\theta}),$$

$$\tilde{\boldsymbol{y}}_j \sim \mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j) := \begin{cases} \mathscr{P}(\boldsymbol{x}_j) & \text{if } w_j = 0 \\ \delta_{\boldsymbol{y}_j^\star} & \text{if } w_j > 0 \end{cases}, \qquad \boldsymbol{w} \in \mathbb{R}_+^{n_u},$$

where $\delta_{\boldsymbol{y}_j^\star}$ denotes the distribution that $\tilde{\boldsymbol{y}}_j$ can only be $\boldsymbol{y}_j^\star$. Therefore, by the definition of $\mathscr{P}_{\boldsymbol{w}}$, we have

$$\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j)}\left[q\left(\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)]\right)\right]=0, \qquad \text{if } w_j>0.$$

Plugging the above definitions into $\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L}_{\boldsymbol{w}}-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}_{\boldsymbol{w}}])]$ we have

$$\begin{aligned}\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L}_{\boldsymbol{w}}-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}_{\boldsymbol{w}}])]&=\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}\left[q\left(\frac{1}{n_u}\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}w_j\left(\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)]\right)\right)\right]\\&=\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}\left[q\left(\frac{1}{n_u}\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\mathbf{1}(w_j>0)w_j\left(\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)]\right)\right)\right]=0.\end{aligned} \tag{20}$$

Therefore, we only need to care about the $\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L}-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}])]$.

$$\begin{aligned}\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[q(\tilde{L}-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}[\tilde{L}])]&=\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}}\left[q\left(\frac{1}{n_u}\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)]\right)\right]\\&\le\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\frac{1}{n_u}\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j)}\left[q\left(\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)-\mathbb{E}_{\mathscr{P}_{\boldsymbol{w}}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)]\right)\right]\\&=\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\mathbf{1}(w_j=0)\frac{1}{n_u}\mathbb{E}_{\mathscr{P}}\left[q\left(\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)-\mathbb{E}_{\mathscr{P}(\boldsymbol{x}_j)}[\ell(\boldsymbol{x}_j,\tilde{\boldsymbol{y}}_j;\cdot)]\right)\right]\\&=\sum_{\boldsymbol{x}_j\in\mathcal{D}_u}\mathbf{1}(w_j=0)\cdot\sigma_j,\end{aligned} \tag{21}$$

where the inequality is by the triangle inequality and the absolute homogeneity of $q(\cdot)$ (Proposition 2.1). Combining equation 20 and equation 21, we have the proposition proved. □

C Omitted Algorithms

In this section we present the two sub-procedures, *i.e.*, line search and de-bias, shared by two main optimization algorithms (Algorithm 2&3). The line search sub-procedure (Algorithm 4) optimally solve the problem of $\arg\min_{\mu\in\mathbb{R}} f_1(\boldsymbol{w}-\mu\boldsymbol{u})$, *i.e.*, given a direction $\boldsymbol{u}$ what is the best step size to move the $\boldsymbol{w}$ along $\boldsymbol{u}$. The de-bias sub-procedure (Algorithm 5) adjusts a sparse $\boldsymbol{w}$ in its own sparse support for a better solution.

Algorithm 4: LineSearch($\boldsymbol{u},\boldsymbol{w}$)

Input: direction $\boldsymbol{u}$; starting point $\boldsymbol{w}$.
Output: step size μ.

$\mu \leftarrow \frac{\langle\Phi\boldsymbol{w}-\boldsymbol{v},\Phi\boldsymbol{u}\rangle+\beta\langle\boldsymbol{w}-\mathbf{1},\boldsymbol{u}\rangle}{\|\Phi\boldsymbol{u}\|_2^2+\beta\|\boldsymbol{u}\|_2^2}$ *(optimal μ)*

Return: μ

Algorithm 5: De-bias($\boldsymbol{w}$)

Input: starting point $\boldsymbol{w}$.
Output: improved $\boldsymbol{w}$.

$\boldsymbol{u} \leftarrow [\nabla f_1(\boldsymbol{w})]_{\text{supp}(\boldsymbol{w})}$ *(in-support grad.)*
$\mu \leftarrow$ LineSearch($\boldsymbol{u},\boldsymbol{w}$)
$\boldsymbol{w} \leftarrow \boldsymbol{w}-\mu\boldsymbol{u}$ *(in-support adjustment)*

Return: $\boldsymbol{w}$

D Implementation Details

All experiments are written in PyTorch 1.8.1 and trained on Tesla V100-SXM GPU. All hyper-parameters are chosen to ensure models achieve good and stable performance on each dataset, and they are kept identical for all active learning approaches.

D.1 Bayesian Active Learning Experiment

Model Architecture We use the exact same model as [16], it is a Bayesian neural network consisting of a ResNet-18 [33] feature extractor followed by a fully connected layer with a ReLU activation, and a final layer allows sampling by local reparametrization [44] with a softmax activation.

Optimization and Hyperparameter Selection Due to larger models and more complicated classification tasks, e.g., CIFAR-100, data augmentation(including random cropping and random horizontal flipping) and learning rate scheduler are used in this experiment to achieve good model performance. The model is optimized with the Adam [45] optimizer using default exponential decay rates (0.9, 0.999) for the moment estimates. Table 1 shows the hyper-parameters in experiment on Bayesian batch active learning, where bs denotes the batch size in dataloader during the model training, lr denotes the learning rate, and wd denotes the weight decay. The hyper-parameters are chosen through grid search.

Table 1: Hyperparameters used in Bayesian active learning experiment

Dataset	Method	Epoch	bs	α	β	lr	wd
Fashion MNIST	SABAL-IHT	200	256	1	10^{-3}	0.001	5×10^{-4}
Fashion MNIST	SABAL-Greedy	200	256	2	0.5	0.001	5×10^{-4}
CIFAR-10	SABAL-IHT	200	256	1	10^{-6}	0.001	5×10^{-4}
CIFAR-10	SABAL-Greedy	200	256	2	1	0.001	5×10^{-4}
CIFAR-100	SABAL-IHT	200	256	1	10^{-6}	0.001	5×10^{-4}
CIFAR-100	SABAL-Greedy	200	256	1	0.5	0.001	5×10^{-4}

D.2 General Active Learning Experiment

Model Architecture On MNIST dataset, we use LeNet-5 model [27], each of the last two fully connected layers followed by dropout layers with dropout probability 0.3. On SVHN and CIFAR10 datasets, we use VGG-16 model [36], each of the last two fully connected layers followed by dropout layers with dropout probability 0.3. For these dropout layers, we implement a consistent MC Dropout to keep the dropout mask consistent for sampling the model parameter.

Optimization and Hyperparameter Selection All models are trained using the cross entropy loss with SGD optimizer, and no data augmentation or learning rate scheduler is used. Tabel 2 shows the hyper-parameters in experiment on general batch active learning, where bs denotes the batch size in dataloader during the model training, lr denotes the learning rate, m denotes the momentum, and wd denotes the weight decay. The hyper-parameters are chosen through grid search.

Table 2: Hyperparameters used in general active learning experiment

Dataset	Method	Epoch	bs	α	β	lr	m	wd
MNIST	SABAL-IHT	150	32	10^{-8}	10^{-3}	0.01	0.9	5×10^{-4}
MNIST	SABAL-Greedy	150	32	10^{-8}	10^{-1}	0.01	0.9	5×10^{-4}
SVHN	SABAL-IHT	150	128	10^{-7}	1	0.01	0.9	5×10^{-4}
SVHN	SABAL-Greedy	150	128	10^{-5}	10^{-1}	0.01	0.9	5×10^{-4}
CIFAR-10	SABAL-IHT	100	128	10^{-3}	10^{-3}	0.001	0.9	5×10^{-4}
CIFAR-10	SABAL-Greedy	100	128	10^{-6}	10^{-6}	0.001	0.9	5×10^{-4}