Black-Box Batch Active Learning for Regression

Anonymous authors
Paper under double-blind review

Abstract

Batch active learning is a popular approach for efficiently training machine learning models on large, initially unlabelled datasets, which repeatedly acquires labels for a batch of data points. However, many recent batch active learning methods are *white-box approaches* limited to differentiable parametric models: they score unlabeled points using acquisition functions based on model embeddings or first- and second-order derivatives. In this paper, we propose *black-box batch active learning* for regression tasks as an extension of white-box approaches. This approach is compatible with a wide range of machine learning models including regular and Bayesian deep learning models and non-differentiable models such as random forests. It is rooted in Bayesian principles and utilizes recent kernel-based approaches. Importantly, our method only relies on model predictions. This allows us to extend a wide range of existing state-of-the-art white-box batch active learning methods (BADGE, BAIT, LCMD) to black-box models. We demonstrate the effectiveness of our approach through extensive experimental evaluations on regression datasets, achieving surprisingly strong performance compared to white-box approaches for deep learning models.

1 Introduction

By selectively acquiring labels for a subset of available unlabelled data, active learning (Cohn et al., 1994) is suited for situations where the acquisition of labels is costly or time-consuming, such as in medical imaging or natural language processing. However, in deep learning, many recent batch active learning methods have focused on *white-box approaches* that rely on the model being parametric and differentiable and which use first or second-order derivatives (e.g. model embeddings)[1].

This can present a limitation in real-world scenarios where model internals or gradients might not be accessible—or might be expensive to access. This is particularly true in the case of 'foundation models' (Bommasani et al., 2021) and large language models such as GPT-3 (Brown et al., 2020), for example, when accessed via a third party. More generally, a lack of differentiability might hinder application of white-box batch active learning approaches to non-differentiable models.

To address these limitations, we propose *black-box batch active learning (B^3AL)* for regression which is compatible with a wider range of machine learning models. Our approach is rooted in Bayesian principles and only requires model predictions from a small (bootstrapped) ensemble. Specifically, we utilize an *(empirical) predictive covariance kernel* based on sampled predictions. We show that the well-known gradient kernel can be seen as an approximation of this predictive covariance kernel.

The proposed approach extends to non-differentiable models through a Bayesian view on the hypothesis space formulation of active learning, based on the ideas behind query-by-committee (Seung et al., 1992). This enables us to use batch active learning methods, such as BAIT (Ash et al., 2021) and BADGE (Ash et al., 2019) in a black-box setting with non-differentiable models, such random forests or gradient-boosted trees.

We evaluate black-box batch active learning on a diverse set of regression datasets. Unlike the above mentioned *white-box* parametric active learning methods which scale in the number of (last-layer) model parameters or the embedding size, our method scales in the number of drawn predictions, and we show that we can

[1] Model embeddings can also be seen as first-order derivatives of the model score under regression in regard to the last layer.

already obtain excellent results with a small ensemble. Our results demonstrate the effectiveness of B^{3}AL in improving the performance of various machine learning models. For deep learning models, B^{3}AL (using an ensemble of size 10) even performs better than the corresponding state-of-the-art white-box methods.

In summary, by leveraging the strengths of kernel-based methods and Bayesian principles, our approach improves the labeling efficiency of a range of differentiable and non-differentiable machine-learning models with surprisingly strong performance.

The rest of the paper is organized as follows: in §2, we discuss related work in active learning and kernel-based methods. In §3, we describe the relevant background and provide a detailed description of our method. In §4, we detail the experimental setup and implementation and provide the results of our experimental evaluation. Finally, §5 concludes with a discussion of our work and directions for future research.

2 Related Work

Active learning has a rich history in the machine learning community, with its origins dating back to seminal works such as (Cohn et al., 1994; Lindley, 1956; Fedorov, 1972; MacKay, 1992). A comprehensive survey of early active learning methods can be found in Settles (2009), while more recent surveys of contemporary deep learning methods can be found in Ren et al. (2021) and Zhan et al. (2022). Our work focuses on pool-based batch active learning, which involves utilizing a pool of unlabeled data and acquiring labels for batches of points rather than individual points at a time (unlike in stream-based active learning or query synthesis). The main challenge in pool-based batch active learning is the choice of the acquisition function.

Differentiable Models. Many acquisition functions approximate well-known information-theoretic quantities (MacKay, 1992), often by approximating Fisher information implicitly or explicitly. This can be computationally expensive, particularly in deep learning where the number of model parameters can be large—even when using last-layer approximations or assuming a generalized linear model (Kirsch & Gal, 2022). BADGE (Ash et al., 2019) and BAIT (Ash et al., 2021) approximate the Fisher information using last-layer loss gradients or the Hessian, respectively, but still have a computational cost scaling with the number of last layer weights. This also applies to methods using similarity matrices (kernels) based on loss gradients of last-layer weights such as SIMILAR (Kothawade et al., 2022) and PRISM (Kothawade et al., 2022), to name a few. Importantly, all of these approaches require differentiable models.

Non-Differentiable Models. *Query-by-committee (QbC, Seung et al. (1992))* measures the disagreement between different model instances to identify informative samples and has been applied to regression (Krogh & Vedelsby, 1994; Burbidge et al., 2007). Kee et al. (2018) extend QbC to the batch setting with a diversity term based on the distance of data points in input space. Nguyen et al. (2012) show batch active learning for random forests. They train an ensemble of random forests and evaluate the joint entropy of the predictions of the ensemble for batch active learning, which can be seen as a special case of BatchBALD in regression.

BALD. Our work most closely aligns with the BALD-family of Bayesian active learning acquisition functions (Houlsby et al., 2011), which focus on classification tasks, however. The crucial insight of BALD is applying the symmetry of mutual information to compute the expected information gain in prediction space instead of in parameter space. As a result, BALD is a *black-box technique* that only leverages model predictions. The extension of BALD to deep learning and multi-class classification tasks using Monte-Carlo dropout models is presented in Gal et al. (2017), which employs batch active learning by selecting the top-B scoring samples from the pool in each acquisition round. BatchBALD (Kirsch et al., 2019) builds upon BALD by introducing a principled approach for batch active learning and consistent MC dropout to account for correlations between predictions. The estimators utilized by Gal et al. (2017) and Kirsch et al. (2019) enumerate over all classes, leading to a trade-off between combinatorial explosion and Monte-Carlo sampling, which can result in degraded quality estimates as acquisition batch sizes increase. Houlsby et al. (2011), Gal et al. (2017), and Kirsch et al. (2019) have not applied BALD to regression tasks. More recently, Jesson et al. (2021) investigate active learning for regression of causal treatment effect and adopt a sampling distribution based on individual-acquisition scores, a heuristic examined in Kirsch et al. (2021).

Kernel-Based Methods. Holzmüller et al. (2022) examine the previously mentioned methods and unify them using gradient-based kernels. Specifically, they express BALD (Houlsby et al., 2011), BatchBALD

(Kirsch et al., 2019), BAIT (Ash et al., 2021), BADGE (Ash et al., 2019), ACS-FW (Pinsler et al., 2019), and Core-Set (Sener & Savarese, 2017)/FF-Active (Geifman & El-Yaniv, 2017) using kernel-based methods for regression tasks. They also propose a new method, LCMD (largest cluster maximum distance).

Our Contribution. We extend the work of Houlsby et al. (2011) and Holzmüller et al. (2022) by combining the prediction-based approach with the kernel-based formulation. This trivally enables batch active learning on regression tasks using black-box predictions for a wide range of existing batch active learning methods.

3 Methodology

Here, we describe the problem setting and motivate the proposed method.

Problem Setting Our proposed method is inspired by the BALD-family of active learning frameworks (Houlsby et al., 2011) and its extension to batch active learning (Kirsch et al., 2019). In our derviation, we make use of a Bayesian model in the narrow sense that we require some stochastic parameters $\boldsymbol{\Omega}$—the model parameters or bootstrapped training data, for example—with a distribution[2] $\mathrm{p}(\boldsymbol{\omega})$:

$$\mathrm{p}(y, \boldsymbol{\omega} \mid \mathbf{x}) = \mathrm{p}(y \mid \mathbf{x}, \boldsymbol{\omega})\, \mathrm{p}(\boldsymbol{\omega}). \tag{1}$$

Bayesian model averaging (BMA) is performed by marginalizing over $\mathrm{p}(\boldsymbol{\omega})$ to obtain the predictive distribution $\mathrm{p}(y \mid \mathbf{x})$, and Bayesian inference for additional data $\mathcal{D}$ by:

$$\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}) \propto \mathrm{p}(\mathcal{D} \mid \boldsymbol{\omega})\, \mathrm{p}(\boldsymbol{\omega}), \tag{2}$$

where $\mathrm{p}(\mathcal{D} \mid \boldsymbol{\omega})$ is the likelihood of the data given the parameters $\boldsymbol{\Omega}$ and $\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D})$ is the new posterior distribution over $\boldsymbol{\Omega}$. Importantly, the choice of $\mathrm{p}(\boldsymbol{\omega})$ covers ensembles (Breiman, 1996; Lakshminarayanan et al., 2017) as well as models with additional stochastic inputs (Osband et al., 2021) or randomized training data by subsampling of the training set, e.g., bagging (Breiman, 1996).

Pool-based Active Learning assumes access to a pool set of unlabeled data $\mathcal{D}^{\text{pool}} = \{\mathbf{x}_i^{\text{pool}}\}$ and a small initially labeled training set $\mathcal{D}^{\text{train}} = \{(\mathbf{x}_i^{\text{train}}, y_i^{\text{train}})\}$, or $\mathcal{D}^{\text{train}} = \emptyset$. In the batch acquisition setting, we want to repeatedly acquire labels for a subset $\{\mathbf{x}_i^{\text{acq}}\}$ of the pool set of a given acquisition batch size B and add them to the training set $\mathcal{D}^{\text{train}}$. Ideally, we want to select samples that are highly 'informative' for the model. For example, these could be samples that are likely to be misclassified or have a large prediction uncertainty for models trained on the currently available training set $\mathcal{D}^{\text{train}}$. Once, we have chosen such an *acquisition batch* $\{\mathbf{x}_i^{\text{acq}}\}$ of unlabeled data, we acquire labels $\{y_i^{\text{acq}}\}$ for these samples and train a new model on the combined training set $\mathcal{D}^{\text{train}} \cup \{(\mathbf{x}^{\text{acq}}{}_i, y^{\text{acq}}{}_i)\}$ and repeat the process. Cruicial to the success of active learning is the choice of acquisition function $\mathcal{A}(\{\mathbf{x}_i^{\text{acq}}\}; \mathrm{p}(\boldsymbol{\omega}))$ which is a function of the acquisition batch $\{\mathbf{x}_i^{\text{acq}}\}$ and the distribution $\mathrm{p}(\boldsymbol{\omega})$ and which we try to maximize in each acquisition round. It measures the informativeness of an acquisition batch for the current model.

Univariate Regression is a common task in machine learning. We assume that the target y is real-valued ($\in \mathbb{R}$) with homoscedastic Gaussian noise:

$$Y \mid \mathbf{x}, \boldsymbol{\omega} \sim \mathcal{N}(\mu(\mathbf{x}; \boldsymbol{\omega}),\, \sigma_N^2). \tag{3}$$

Equivalently, $Y \mid \mathbf{x}, \boldsymbol{\omega} \sim \mu(\mathbf{x}; \boldsymbol{\omega}) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0,\, \sigma_N^2)$. As usual, we assume that the noise is independent for different inputs $\mathbf{x}$ and parameters $\boldsymbol{\omega}$. Homoscedastic noise is a special case of the general heteroscedastic setting: the noise variance is simply a constant. Our approach in this paper could easily be extended to heteroscedastic noise by replacing σ_N with a function $\sigma_N(\mathbf{x}; \boldsymbol{\omega})$, but for this work we limit ourselves to the simplest case.

3.1 Kernel-based Methods & Information Theory

We build on Holzmüller et al. (2022) which expresses contemporary batch active learning methods using kernel methods. While a full treatment is naturally beyond the scope of this paper, we briefly review some key ideas here. We refer the reader to the extensive paper for more details.

[2]We do not require a prior distribution as active learning is not concerned with how we arrive at the model we want to acquire labels for. We can define a prior and perform, e.g, variational inference, but we do not *need* to. Hence, we use $\mathrm{p}(\boldsymbol{\omega})$.

Gaussian Processes are one way to introduce kernel-based methods. A simple way to think about Gaussian Processes (Williams & Rasmussen, 2006; Lázaro-Gredilla & Figueiras-Vidal, 2010; Rudner et al., 2022) is as Bayesian linear regression model with an implicit, potentially infinite-dimensional feature space (depending on the covariance kernel) that uses the kernel trick to abstract away the feature map which maps the input space to the feature space.

Multivariate Gaussian Distribution. The distinctive property of a Gaussian Process is that all predictions are jointly Gaussian distributed. We can then write the joint distribution for a univariate regression model as:

$$Y_1, \dots, Y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n \sim \mathcal{N}(\mathbf{0}, \operatorname{Cov}[\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_n)] + \sigma_N^2 \mathbf{I}), \tag{4}$$

where $\mu(\mathbf{x})$ are the observation-noise free predictions as random variables and $\operatorname{Cov}[\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_n)]$ is the covariance matrix of the predictions. The covariance matrix is defined via the kernel function $k(\mathbf{x}, \mathbf{x}')$:

$$\operatorname{Cov}[\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_n)] = \Big[k(\mathbf{x}_i, \mathbf{x}_j)\Big]_{i,j=1}^{n}. \tag{5}$$

The kernel function $k(\mathbf{x}, \mathbf{x}')$ can be chosen almost arbitrarily, e.g. see Williams & Rasmussen (2006, Ch. 4). The linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$ and the radial basis function kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2}|\mathbf{x} - \mathbf{x}'|^2)$ are common examples, as is the gradient kernel, which we examine next.

Fisher Information & Linearization. When using neural networks for regression, the gradient kernel

$$k_{\text{grad}}(\mathbf{x}, \mathbf{x}' \mid \boldsymbol{\omega}^*) \triangleq \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}; \boldsymbol{\omega}^*) \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)]^{-1} \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}'; \boldsymbol{\omega}^*)^\top \tag{6}$$

$$= \langle \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}; \boldsymbol{\omega}^*), \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}'; \boldsymbol{\omega}^*) \rangle_{\nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)]^{-1}} \tag{7}$$

is the canonical choice, where $\boldsymbol{\omega}^*$ is a *maximum likelihood or maximum a posteriori estimate (MLE, MAP)* and $\nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)]$ is the Hessian of the prior at $\boldsymbol{\omega}^*$. Note that $\nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}; \boldsymbol{\omega}^*)$ is a *row* vector. Commonly, the prior is a Gaussian distribution with zero mean and a identity covariance matrix, and thus $\nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)] = \mathbf{I}$.

The significance of this kernel lies in its relationship with the Fisher information matrix at $\boldsymbol{\omega}^*$ as outlined in Immer (2020); Immer et al. (2021); Kirsch & Gal (2022), or equivalently, with the linearization of the loss function around $\boldsymbol{\omega}^*$, as demonstrated in Holzmüller et al. (2022). This leads to a Gaussian approximation, which when combined with a Gaussian likelihood results in a Gaussian posterior distribution for predictions. The use of the finite-dimensional gradient kernel results in an implicit Bayesian linear regression in the context of regression models.

Posterior Gradient Kernel. We can use the well-known properties of multivariate normal distribtions to marginalize or condition the joint distribution in (4). Following Holzmüller et al. (2022), this allows us to explicitly obtain the posterior gradient kernel given additional $\mathbf{x}_1, \dots, \mathbf{x}_n$ as:

$$\begin{aligned} & k_{\text{grad}\to\text{post}(\mathbf{x}_1,\dots,\mathbf{x}_n)}(\mathbf{x}, \mathbf{x}' \mid \boldsymbol{\omega}^*) \\ & \triangleq \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}; \boldsymbol{\omega}^*) \left(\sigma_N^{-2} \begin{pmatrix} \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}_1; \boldsymbol{\omega}^*) \\ \vdots \\ \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}_n; \boldsymbol{\omega}^*) \end{pmatrix} \begin{pmatrix} \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}_1; \boldsymbol{\omega}^*) \\ \vdots \\ \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}_n; \boldsymbol{\omega}^*) \end{pmatrix}^\top + \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^*)] \right)^{-1} \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}'; \boldsymbol{\omega}^*)^\top. \end{aligned} \tag{8}$$

The factor σ_N^{-2} originates from implicitly conditioning on the observations $Y_i \mid \mathbf{x}_i$, which include observation noise.

Importantly for active learning, the multivariate normal distribution is the maximum entropy distribution for a given covariance matrix, and is thus an upper-bound for the entropy of any distribution with the same covariance matrix. The entropy is given by the log-determinant of the covariance matrix:

$$\mathrm{H}[Y_1, \dots, Y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n] = \frac{1}{2} \log \det(\operatorname{Cov}[\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_n)] + \sigma^2 \mathbf{I}) + C_n, \tag{9}$$

where $C_n \triangleq \frac{n}{2} \log(2\,\pi\,e)$ is a constant that only depends on the number of samples n. Connecting kernel-based methods to information-theoretic quantities like the expected information gain, we then know that the respective acquisition scores are upper-bounds on the actual expected information gain, as we see below.

Expected Information Gain (EIG) (Lindley, 1956) is based on the information-theoretic principle of mutual information. For a Bayesian model, the EIG between the parameters $\boldsymbol{\Omega}$ and the predictions $Y \mid x$ is defined as:

$$\mathrm{I}[\boldsymbol{\Omega}; Y \mid \mathbf{x}] = \mathrm{H}[\boldsymbol{\Omega}] - \mathrm{H}[\boldsymbol{\Omega} \mid Y, \mathbf{x}] = \int \mathrm{p}(\boldsymbol{\omega}, y \mid \mathbf{x}) \log \frac{\mathrm{p}(\boldsymbol{\omega} \mid y, \mathbf{x})}{\mathrm{p}(\boldsymbol{\omega})} \, d\boldsymbol{\omega} \, dy, \tag{10}$$

where $\mathrm{I}[\boldsymbol{\Omega}; Y \mid \mathbf{x}]$ is the mutual information between the parameters $\boldsymbol{\Omega}$ and the model predictions Y given the data $\mathbf{x}$; $\mathrm{H}[\boldsymbol{\Omega}]$ is the entropy of the parameters $\boldsymbol{\Omega}$; and $\mathrm{H}[\boldsymbol{\Omega} \mid Y, \mathbf{x}]$ is the conditional entropy of the parameters $\boldsymbol{\Omega}$ in expectation over the predictions $\boldsymbol{\Omega}$ and the model predictions $Y \mid \mathbf{x}$. The EIG measures the amount of information that can be gained about the parameters $\boldsymbol{\Omega}$ by observing the predictions $Y \mid \mathbf{x}$ on average (assuming that the model is well-specified), where we think of the entropy $\mathrm{H}[\boldsymbol{\Omega}]$ as capturing the uncertainty about the parameters $\boldsymbol{\Omega}$.

Computing the EIG can be difficult when focused on the model paramters, as we need to compute the conditional entropy $\mathrm{H}[\boldsymbol{\Omega} \mid Y, \mathbf{x}]$—yet this is what many recent approaches effectively attempt to do via the Fisher information (Kirsch & Gal, 2022). (We do not need to compute the entropy of the parameters $\mathrm{H}[\boldsymbol{\Omega}]$, as it does not depend on the data $\mathbf{x}$ and can be dropped from the optimization objective.)

BALD (Houlsby et al., 2011) examines the expected information gain in *prediction space* instead of parameter space, using the symmetry of the mutual information:

$$\mathrm{I}[\boldsymbol{\Omega}; Y \mid \mathbf{x}] = \mathrm{I}[Y; \boldsymbol{\Omega} \mid \mathbf{x}] = \mathrm{H}[Y \mid \mathbf{x}] - \mathrm{H}[Y \mid \boldsymbol{\Omega}, \mathbf{x}] = \int \mathrm{p}(\boldsymbol{\omega}, y \mid \mathbf{x}) \log \frac{\mathrm{p}(y \mid \mathbf{x}, \boldsymbol{\omega})}{\mathrm{p}(y \mid \mathbf{x})} \, d\boldsymbol{\omega} \, dy, \tag{11}$$

This can be much easier to evaluate as the expectations are over the prediction variables Y instead, which can be approximated by sampling from $\boldsymbol{\Omega}$. BatchBALD (Kirsch et al., 2019) is an extension of BALD to batch acquisition of samples $\{\mathbf{x}_i^{\mathrm{acq}}\}$ for classification tasks:

$$\mathrm{I}[\boldsymbol{\Omega}; Y_1, \ldots, Y_B \mid \mathbf{x}_1, \ldots, \mathbf{x}_B] = \mathrm{H}[Y_1, \ldots, Y_B \mid \mathbf{x}_1, \ldots, \mathbf{x}_B] - \sum_i \mathrm{H}[Y_i \mid \mathbf{x}_i, \boldsymbol{\Omega}], \tag{12}$$

using the mutual information between the parameters $\boldsymbol{\omega}$ and the predictions $\{Y_i\}$ on an acquisition candidate batch $\{\mathbf{x}_i\}$, and thus takes the correlations between samples into account.

This joint entropy can be upper-bounded by the corresponding entropy of multivariate normal distribution with the same covariance matrix, while the conditional entropy term is precisely the entropy of a normal distribution given our model assumptions, yielding an upper-bound overall.

The same approach can be applied to other information quantities like the expected predicted information gain. Unlike Kirsch & Gal (2022), which examines the connection between Fisher information (weight-space) and information-theoretic quantities, this exposition shows the connection between predictions and information-theoretic quantities.

Greedy Acquisition is a heuristic to find the subset of size B in $\mathcal{D}^{\mathrm{pool}}$ that maximizes $\mathcal{A}(\{\mathbf{x}_i\}, \mathrm{p}(\boldsymbol{\omega}))$. Finding the optimal solution is NP-hard (Schrijver et al., 2003). Kirsch et al. (2019) find that $\mathcal{A}_{\mathrm{BatchBALD}}$ is submodular and that iteratively adding the sample $\mathbf{x}^{\mathrm{acq}}{}_i$ with the highest $\mathcal{A}_{\mathrm{BatchBALD}}(\mathbf{x}^{\mathrm{acq}}{}_1, \ldots, \mathbf{x}^{\mathrm{acq}}{}_i, \mathrm{p}(\boldsymbol{\omega}))$ to the acquisition batch size is a $1 - 1/e$-approximation. This is a well-known result in submodular optimization and is known as the *greedy algorithm* (Nemhauser et al., 1978). When viewed through the entropy approximation using the log determinant, finding an optimal solution is equivalent to finding the mode of the k-DPP (Kulesza & Taskar, 2011).

Other Information-Theoretic Quantities and Selection Methods. EPIG (Bickford Smith et al., 2023) and other information-theoretic quantities can be examind similarly. We refer to Kirsch & Gal (2022) for more details. For other selection methods, we refer to Holzmüller et al. (2022).

3.2 Black-Box Batch Active Learning

We formally introduce the prediction-based covariance kernel and compare it to the gradient kernel commonly used for active learning with deep learning models in parameter-space. For non-differentiable models, we show how it can also be derived using a Bayesian model perspective on hypotheses.

In addition to being illustrative, this section allows us to connect prediction-based kernels to the kernels used by Holzmüller et al. (2022), which in turns connects them to various SotA active learning methods.

3.2.1 Prediction Kernel

To perform black-box batch active learning, we directly use the *predictive covariance* of $Y_i|\mathbf{x}_i$ and $Y_j|\mathbf{x}_j$:

$$\mathrm{Cov}_{\boldsymbol{\Omega}}[Y_i; Y_j \mid \mathbf{x}_i, \mathbf{x}_j] = \mathrm{Cov}_{\boldsymbol{\Omega}}[\mu^{\boldsymbol{\omega}}_{\mathbf{x}_i}; \mu^{\boldsymbol{\omega}}_{\mathbf{x}_j}] + \sigma_N^2 \mathbb{1}\{i = j\}. \tag{13}$$

where we have abbreviated $\mu(\mathbf{x}; \boldsymbol{\omega})$ with $\mu^{\boldsymbol{\omega}}_{\mathbf{x}}$, and used the law of total covariance and the fact that the noise is uncorrelated between samples.

We define the *predictive covariance kernel* $k_{\mathrm{pred}}(\mathbf{x}_i, \mathbf{x}_j)$ as the covariance of the predicted means:

$$k_{\mathrm{pred}}(\mathbf{x}_i; \mathbf{x}_j) \triangleq \mathrm{Cov}_{\boldsymbol{\Omega}}[\mu^{\boldsymbol{\omega}}_{\mathbf{x}_i}; \mu^{\boldsymbol{\omega}}_{\mathbf{x}_j}]. \tag{14}$$

Unlike in §3.1, we do not define the covariance via the kernel but the kernel via the covariance.

3.2.2 Empirical Predictive Covariance Kernel

For sampled (empirical) model parameters $\hat{\boldsymbol{\Omega}} = (\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_K) \sim \mathrm{p}(\boldsymbol{\omega})$, we define the *empirical predictive covariance kernel* $k_{\widehat{\mathrm{pred}}}(\mathbf{x}_i; \mathbf{x}_j)$:

$$k_{\widehat{\mathrm{pred}}}(\mathbf{x}_i; \mathbf{x}_j) \triangleq \widehat{\mathrm{Cov}}_{\boldsymbol{\Omega}}[\mu^{\boldsymbol{\omega}}_{\mathbf{x}_i}; \mu^{\boldsymbol{\omega}}_{\mathbf{x}_j}] = \frac{1}{K} \sum_{k=1}^{K} \left(\mu^{\boldsymbol{\omega}_k}_{\mathbf{x}_i} - \frac{1}{K} \sum_{l=1}^{K} \mu^{\boldsymbol{\omega}_l}_{\mathbf{x}_i} \right) \left(\mu^{\boldsymbol{\omega}_k}_{\mathbf{x}_j} - \frac{1}{K} \sum_{l=1}^{K} \mu^{\boldsymbol{\omega}_l}_{\mathbf{x}_j} \right) \tag{15}$$

$$= \left\langle \frac{1}{\sqrt{K}} (\bar{\mu}^{\boldsymbol{\omega}_1}_{\mathbf{x}_i}, \dots, \bar{\mu}^{\boldsymbol{\omega}_K}_{\mathbf{x}_i}), \frac{1}{\sqrt{K}} (\bar{\mu}^{\boldsymbol{\omega}_1}_{\mathbf{x}_j}, \dots, \bar{\mu}^{\boldsymbol{\omega}_K}_{\mathbf{x}_j}) \right\rangle, \tag{16}$$

with centered predictions $\bar{\mu}^{\boldsymbol{\omega}_k}_{\mathbf{x}} \triangleq \mu^{\boldsymbol{\omega}_k}_{\mathbf{x}} - \frac{1}{K} \sum_{l=1}^{K} \mu^{\boldsymbol{\omega}_l}_{\mathbf{x}}$.

3.2.3 Differentiable Models

Similar to Holzmüller et al. (2022, §C.1), we show that the posterior gradient kernel is a first-order approximation of the covariance kernel. This section explicitly conditions on $\mathcal{D}^{\mathrm{train}}$. The result is simple but instructive:

Proposition 3.1. *The* posterior *gradient kernel* $k_{\mathrm{grad} \to \mathrm{post}(\mathcal{D}^{\mathrm{train}})}(\mathbf{x}_i; \mathbf{x}_j \mid \boldsymbol{\omega}^*)$ *is an approximation of the predictive covariance kernel* $k_{\mathrm{pred}}(\mathbf{x}_i; \mathbf{x}_j)$.

Proof. We use a first-order Taylor expansion of the mean function $\mu(\mathbf{x}; \boldsymbol{\omega})$ around $\boldsymbol{\omega}^*$:

$$\mu(\mathbf{x}; \boldsymbol{\omega}) \approx \mu(\mathbf{x}; \boldsymbol{\omega}^*) + \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}; \boldsymbol{\omega}^*) \underbrace{(\boldsymbol{\omega} - \boldsymbol{\omega}^*)}_{\triangleq \Delta\boldsymbol{\omega}}. \tag{17}$$

Choose $\boldsymbol{\omega}^* = \mathbb{E}_{\boldsymbol{\omega} \sim \mathrm{p}(\boldsymbol{\omega} | \mathcal{D}^{\mathrm{train}})}[\boldsymbol{\omega}]$ (BMA). Then we have $\mathbb{E}_{\mathrm{p}(w | \mathcal{D}^{\mathrm{train}})}[\mu(\mathbf{x}; \boldsymbol{\omega})] = \mu(\mathbf{x}; \boldsymbol{\omega}^*)$. We then have:

$$k_{\mathrm{pred}}(\mathbf{x}_i; \mathbf{x}_j) = \mathrm{Cov}_{\boldsymbol{\omega} sim\, \mathrm{p}(\boldsymbol{\omega} | \mathcal{D}^{\mathrm{train}})}[\mu(\mathbf{x}_i; \boldsymbol{\omega}); \mu(\mathbf{x}_j; \boldsymbol{\omega})] \tag{18}$$

$$\approx \mathbb{E}_{\boldsymbol{\omega}^* + \Delta\boldsymbol{\omega} \sim \mathrm{p}(w | \mathcal{D}^{\mathrm{train}})}[\langle \nabla_{\boldsymbol{\omega}} \mu^{\boldsymbol{\omega}^*}_{\mathbf{x}_i} \Delta\boldsymbol{\omega}, \nabla_{\boldsymbol{\omega}} \mu^{\boldsymbol{\omega}^*}_{\mathbf{x}_j} \Delta\boldsymbol{\omega} \rangle] \tag{19}$$

$$= \nabla_{\boldsymbol{\omega}} \mu_{\mathbf{x}_i}^{\boldsymbol{\omega}^*} \; \mathbb{E}_{\boldsymbol{\omega}^* + \Delta\boldsymbol{\omega} \sim \mathrm{p}(w \mid \mathcal{D}^{\text{train}})} [\Delta\boldsymbol{\omega} \Delta\boldsymbol{\omega}^\top] \, \nabla_{\boldsymbol{\omega}} \mu_{\mathbf{x}_j}^{\boldsymbol{\omega}^*}{}^\top \tag{20}$$

$$= \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}_i; \boldsymbol{\omega}^*) \; \mathrm{Cov}[\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}}] \, \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}_j; \boldsymbol{\omega}^*)^\top \tag{21}$$

$$\approx k_{\text{grad} \to \text{post}(\mathcal{D}^{\text{train}})}(\mathbf{x}_i; \mathbf{x}_j \mid \boldsymbol{\omega}^*). \tag{22}$$

The intermediate expectation is the model covariance $\mathrm{Cov}[\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}}]$ as $\boldsymbol{\omega}^*$ is the BMA. For the last step, we use the Gauss-Newton approximation (Immer et al., 2021; Kirsch & Gal, 2022) and approximate the inverse of the covariance using the Hessian of the negative log likelihood at $\boldsymbol{\omega}^*$:

$$\mathrm{Cov}[\boldsymbol{\Omega} \mid \mathcal{D}^{\text{train}}]^{-1} \approx \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\boldsymbol{\omega}^* \mid \mathcal{D}^{\text{train}})] \tag{23}$$

$$= \nabla^2_{\boldsymbol{\omega}} [-\log \mathrm{p}(\mathcal{D}^{\text{train}} \mid \boldsymbol{\omega}^*) - \log \mathrm{p}(\boldsymbol{\omega}^*)] \tag{24}$$

$$= \sigma^{-2} \textstyle\sum_i \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}^{\text{train}}{}_i; \boldsymbol{\omega}^*)^\top \nabla_{\boldsymbol{\omega}} \mu(\mathbf{x}^{\text{train}}{}_i; \boldsymbol{\omega}^*) \tag{25}$$

$$- \nabla^2_{\boldsymbol{\omega}} \log \mathrm{p}(\boldsymbol{\omega}^*), \tag{26}$$

where we have first used Bayes' theorem and that $\mathrm{p}(\mathcal{D}^{\text{train}})$ vanishes under differentiation—it is constant in $\boldsymbol{\omega}$. Secondly, the Hessian of the negative log likelihood is just the outer product of the gradients divided by the noise variance in the homoscedastic regression case. $\nabla^2_{\boldsymbol{\omega}}[-\log \mathrm{p}(\boldsymbol{\omega}^*)]$ is the prior term. This matches (8). □

3.2.4 Non-Differentiable Models

How can we apply the above result to non-differentiable models? In the following, we use a Bayesian view on the hypothesis space to show that we can connect the empirical predictive covariance kernel to a gradient kernel, too. With $\hat{\boldsymbol{\Omega}}$ as defined previously, we introduce a latent Ψ to represent the 'true' hypothesis $\boldsymbol{\omega}_\psi \in \hat{\boldsymbol{\Omega}}$ from the empirical hypothesis space $\hat{\boldsymbol{\Omega}}$. This is similar to QbC (Seung et al., 1992).

Specifically, we use a one-hot categorical distribution, that is a multinomial distribution from which we draw one sample. We use $\Psi \sim \mathrm{Multinomial}(\boldsymbol{q}, 1)$ over the empirical $\boldsymbol{\omega}_k$ as hypotheses, with $\boldsymbol{q} \in S^{K-1}$ defining the distribution; S^{K-1} the $K-1$ simplex in $\mathbb{R}^K$; $\boldsymbol{q}_k = \mathrm{p}(\boldsymbol{\omega}_\psi = k)$; and $\sum_{k=1}^K \boldsymbol{q}_k = 1$. For the predictive $\tilde{\mu}(\mathbf{x}; \psi)$, we have:

$$\tilde{\mu}(\mathbf{x}; \psi) \triangleq \mu(\mathbf{x}; \boldsymbol{\omega}_\psi) = \langle \mu(\mathbf{x}; \cdot), \psi \rangle, \tag{27}$$

where $\mu(\mathbf{x}; \cdot)$ is a column vector of the predictions $\mu(\mathbf{x}; \boldsymbol{\omega}_k)$ for $\mathbf{x}$ for all $\boldsymbol{\omega}_k$.

We now examine this model and its kernels. We choose $\boldsymbol{q}$ as a uninformative[3] uniform distribution over the hypotheses $\boldsymbol{q} = \frac{1}{K}$. The BMA then matches the empirical mean:

$$\tilde{\mu}(\mathbf{x}; \boldsymbol{q}) \triangleq \mathbb{E}_{\mathrm{p}(\psi \mid \boldsymbol{q})}[\mu(\mathbf{x}; \boldsymbol{\omega}_\psi)] = \langle \mu(\mathbf{x}; \cdot), \boldsymbol{q} \rangle = \sum_{\psi=1}^{K} \boldsymbol{q}_\psi \mu(\mathbf{x}; \boldsymbol{\omega}_\psi) = \sum_{\psi=1}^{K} \frac{1}{K} \mu(\mathbf{x}; \boldsymbol{\omega}_\psi). \tag{28}$$

What is the predictive covariance kernel of this model? And what is the posterior gradient kernel for $\boldsymbol{q}$?

Proposition 3.2. *We have:*

1. *The predictive covariance kernel $k_{\text{pred},\psi}(\mathbf{x}_i, \mathbf{x}_j)$ for $\hat{\boldsymbol{\Omega}}$ using uniform $\boldsymbol{q}$ is equal to the empirical predictive covariance kernel $k_{\widehat{\text{pred}}}(\mathbf{x}_i; \mathbf{x}_j)$.*

2. *The gradient kernel $k_{\text{grad},\boldsymbol{q} \to \text{post}(\mathcal{D}^{\text{train}})}(\mathbf{x}_i; \mathbf{x}_j)$ for $\hat{\boldsymbol{\Omega}}$ in respect to uniform $\boldsymbol{q}$ is equal to the empirical predictive covariance kernel $k_{\widehat{\text{pred}}}(\mathbf{x}_i; \mathbf{x}_j)$.*

Proof. Like for the previous differentiable model, the BMA of the model parameters Ψ is just $\boldsymbol{q}$: $\mathbb{E}\,\Psi = \boldsymbol{q}$. The first statement immediately follows:

$$k_{\text{pred},\psi}(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{Cov}_\psi[\tilde{\mu}(\mathbf{x}_i; \psi); \tilde{\mu}(\mathbf{x}_j; \psi)] = \mathbb{E}_{\mathrm{p}(\psi)}[\bar{\mu}_{\mathbf{x}_i}^{\boldsymbol{\omega}_\psi} \, \bar{\mu}_{\mathbf{x}_j}^{\boldsymbol{\omega}_\psi}] = \tfrac{1}{K} \sum_\psi \bar{\mu}_{\mathbf{x}_i}^{\boldsymbol{\omega}_\psi} \bar{\mu}_{\mathbf{x}_j}^{\boldsymbol{\omega}_\psi} = k_{\widehat{\text{pred}}}(\mathbf{x}_i; \mathbf{x}_j). \tag{29}$$

[3] If we had additional information about the $\hat{\boldsymbol{\Omega}}$—for example, if we had validation set losses—we could use this information to set $\boldsymbol{q}$.

For the second statement, we begin with the same linearization as in the previous proposition (but this time in respect to $\boldsymbol{q}$):

$$k_{\text{grad},\boldsymbol{q}\to\text{post}(\mathcal{D}^{\text{train}})}(\mathbf{x}_i;\mathbf{x}_i) = \text{Cov}_{\psi\sim\text{p}(\psi)}[\mu(\mathbf{x}_i;\psi);\mu(\mathbf{x}_j;\psi)] \tag{30}$$

$$= \mathbb{E}_{\boldsymbol{q}+\Delta\Psi\sim\text{p}(\psi)}[\langle\nabla_{\boldsymbol{q}}\tilde{\mu}(\mathbf{x}_i;\boldsymbol{q})\Delta\Psi, \nabla_{\boldsymbol{q}}\tilde{\mu}(\mathbf{x}_j;\boldsymbol{q})\Delta\Psi\rangle] \tag{31}$$

$$= \nabla_{\boldsymbol{q}}\tilde{\mu}(\mathbf{x}_i;\boldsymbol{q})\ \text{Cov}[\Psi]\ \nabla_{\boldsymbol{q}}\tilde{\mu}(\mathbf{x}_i;\boldsymbol{q})^\top. \tag{32}$$

We can read off a linearization for $\nabla_{\boldsymbol{q}}\tilde{\mu}(\mathbf{x}_i;\boldsymbol{q})$ from the inner product in Equation (28):

$$\nabla_{\boldsymbol{q}}\tilde{\mu}(\mathbf{x}_i;\boldsymbol{q}) = \mu(\mathbf{x};\cdot)^\top. \tag{33}$$

Lastly, the covariance of the multinomial Ψ is:

$$\text{Cov}[\Psi] = \text{diag}(\boldsymbol{q}) - \boldsymbol{q}\boldsymbol{q}^\top. \tag{34}$$

Putting this all together:

$$k_{\text{grad},\boldsymbol{q}\to\text{post}(\mathcal{D}^{\text{train}})}(\mathbf{x}_i;\mathbf{x}_i) = \mu(\mathbf{x}_i;\cdot)^\top\,\text{diag}(\boldsymbol{q})\,\mu(\mathbf{x}_j;\cdot) - (\mu(\mathbf{x}_i;\cdot)^\top\,\boldsymbol{q})\,(\boldsymbol{q}^\top\,\mu(\mathbf{x}_j;\cdot)) \tag{35}$$

$$= \frac{1}{K}\sum_{\psi}\mu(\mathbf{x}_i;\boldsymbol{\omega}_\psi)\,\mu(\mathbf{x}_j;\boldsymbol{\omega}_\psi)^\top - \left(\frac{1}{K}\sum_{\psi}\mu(\mathbf{x}_i;\boldsymbol{\omega}_\psi)\right)\left(\frac{1}{K}\sum_{\psi}\mu(\mathbf{x}_j;\boldsymbol{\omega}_\psi)\right) = k_{\widehat{\text{pred}}}(\mathbf{x}_i;\mathbf{x}_j). \tag{36}$$

□

Above gradient kernel is only the posterior gradient kernel in the sense that we have sampled $\boldsymbol{\omega}_\psi$ from the non-differentiable model after inference on training data. The samples themselves were drawn uniformly.

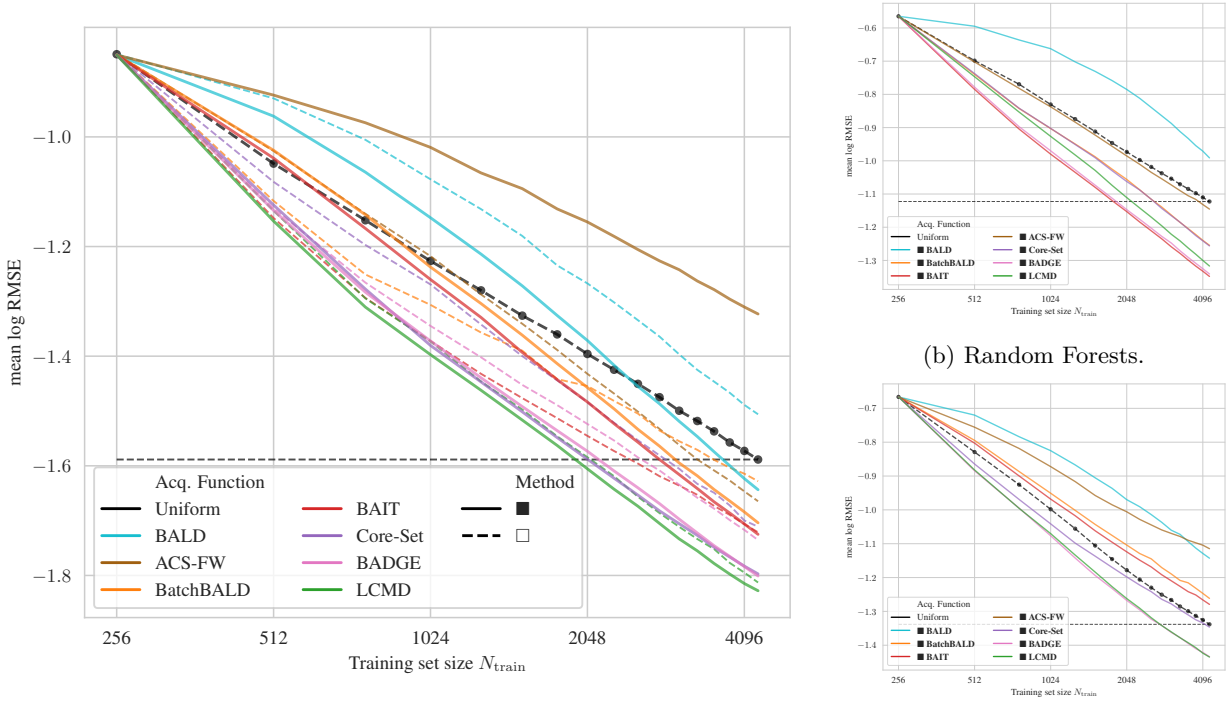
A similar Bayesian model using Bayesian Model Combination (BMC) could be set up which allows for arbitrary convex mixtures of the ensemble members. This would entail using a Dirichlet distribution Dirichlet($\boldsymbol{\alpha}$) instead of the multinomial distribution. Assuming an uninformative prior, this leads to the same results up to a factor of $1 + \sum_k \boldsymbol{\alpha}_k$.

This demonstrates that a straightforward Bayesian model can be constructed on top of a non-differentiable ensemble model. Bayesian inference in this context aims to identify the most suitable member of the ensemble. Given the limited number of samples and likelihood of model misspecification, it is likely that none of the members accurately represents the true model. However, for active learning purposes, the main focus is solely on quantifying the degree of disagreement among the ensemble members.

Application to DNNs, BNNs, and Other Models. The proposed approach has relevance due to its versatility, as it can be applied to a wide range of models that can be consistently queried for prediction, including deep ensembles (Lakshminarayanan et al., 2016), Bayesian neural networks (BNNs) (Blundell et al., 2015; Gal & Ghahramani, 2015), and non-differentiable models. The kernel used in this approach is simple to implement and scales in the number of empirical predictions per sample, rather than in the parameter space, as seen in other methods such as Ash et al. (2021).

4 Results

We follow the evaluation from Holzmüller et al. (2022) and use their framework and the same experimental setup to ease comparison. This allows us to directly compare to several SotA methods in a regression setting, respectively their kernel-based analogues. Specifically, we compare to the following popular deep active learning methods: BALD (Houlsby et al., 2011), BatchBALD (Kirsch et al., 2019), BAIT (Ash et al., 2021), BADGE (Ash et al., 2019), ACS-FW (Pinsler et al., 2019), Core-Set (Sener & Savarese, 2017)/FF-Active (Geifman & El-Yaniv, 2017), and LCMD (Holzmüller et al., 2022). We also compare to the random selection baseline ('Uniform'). We use 15 large tabular datasets from the UCI Machine Learning Repository (Dua & Graff, 2017) and the OpenML benchmark suite (Vanschoren et al., 2014) for our experiments, see Table 3.



(a) Deep Neural Networks.

(b) Random Forests.

(c) Gradient-Boosted Trees.

Figure 1: *Mean logarithmic RMSE over 15 regression datasets.* **(a)** For DNNs, we see that black-box ■ methods (using ensembles) work as well as white-box □ methods, and in most cases better, with the exception of ACS-FW. **(b)** For random forests with the default hyperparameters from scikit-learn (Pedregosa et al., 2011), we see that black-box methods perform better than the uniform baseline, with the exception of BALD, which uses top-k acquisition. **(c)** For gradient-boosted trees (Dorogush et al., 2018) with a *virtual ensemble*, we hypothesize that the virtual ensembles do not capture predictive disagreement well enough for active learning. In the appendix, see Table 1 for average performance metrics.

Experimental Setup. Our experiments ran on A100 GPUs with 40GB of GPU memory. We used the same experimental setup as Holzmüller et al. (2022) and used the same hyperparameters for all methods.

Ensemble Size. For deep learning, we use a small ensemble of 10 models, which is sufficient to achieve good performance. This is 'small' in the regression setting—cf. TK citation TK uses 20 models for comparison, whereas in Bayesian Optimal Experiment Design much higher sample counts are regularly used. In many cases, training an ensemble of regression models is still fast and cheap compared to the cost of acquiring additional labels. For non-differentiable models, we experiment with random forests (Breiman, 2001), using the different trees as ensemble members, and a virtual ensemble of gradient-boosted decision trees (Prokhorenkova et al., 2017). For random forests, we use the implementation provided in scikit-learn (Pedregosa et al., 2011) with default hyperparameters, that is using 100 trees per forest. For gradient-boosted decision trees, we use a virtual ensemble of up to 20 with early stopping using a validation set[4]. We use the implementation in CatBoost (Dorogush et al., 2018). We do not perform any hyperparameter tuning.

Black-Box vs White-Box Deep Active Learning. In Figure 1a and 2, we see that B^{3}AL is competitive with white-box active learning, when using BALD, BatchBALD, BAIT, BADGE, and ACS-FW. On average, it even outperforms the white-box methods on the 15 datasets we analyzed (excluding ACS-FW and BAIT). We hypothesize that this is due to the implicit Fisher information approximation in the white-box methods (Kirsch & Gal, 2022), which is not as accurate as the explicit approximation in B^{3}AL, in the low data regime.

[4]If the virtual ensemble creation fails because there are no sufficiently many trees due to early stopping, we halve the ensemble size and retry. This was only needed for the *poker* dataset.

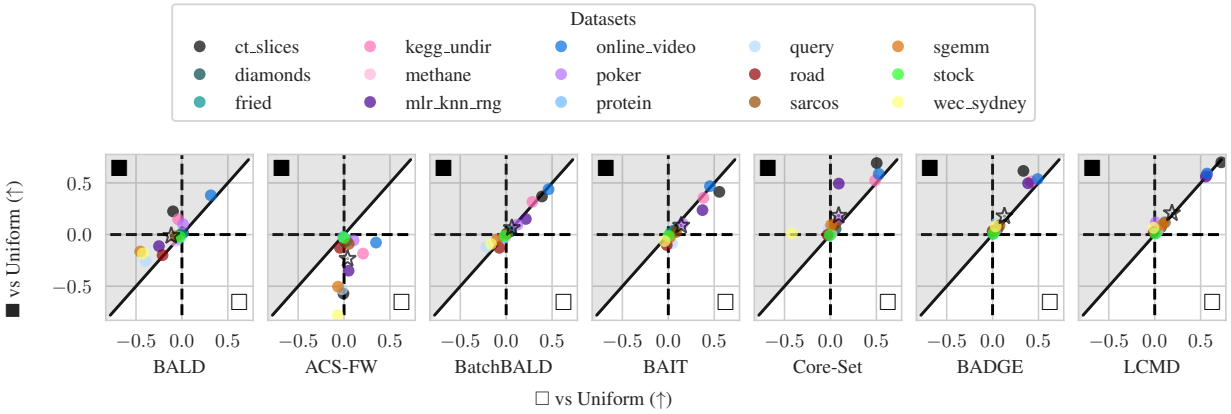


Figure 2: *Average Logarithmic RMSE by regression datasets for DNNs:* ■ *vs* □ *(vs Uniform).* Across acquisition functions, the performance of black-box methods is highly correlated with the performance of white-box methods, even though black-box methods make fewer assumptions about the model. We plot the improvement of the white-box □ method over the uniform baseline on the x-axis, so for datasets with markers left of the dashed vertical lines, the white-box method performs better than uniform, and the improvement of the black-box ■ method over the uniform baseline on the y-axis, so for datasets with markers above the dashed horizontal lines, the black-box method performs better than uniform. Similarly, for datasets with markers in the ■ region, the black-box method performs better than the white-box method. The average over all datasets is marked with a star ⋆. Surprisingly, on average over all acquisition rounds, the black-box methods perform better than the white-box methods for all but ACS-FW and BAIT. In the appendix, see Figure 3 for the final acquisition round only and Table 3 for details on the datasets.

On the other hand, it seems that the black-box methods suffer from the low feature space dimensionality, as they are much closer to BALD.

Non-Differentiable Models. In Figure 1b and c, see that B^{3}AL also works well for non-differentiable models, such as random forests and gradient-boosted decision trees. While for random forests, all methods except for BALD (as top-k selection method), outperform uniform acquisition, for gradient-boosted decision trees, B^{3}AL only works better than random acquisition when using LMCD and BADGE. This might be because of the virtual ensemble.

5 Conclusion and Future Work

In this paper, we have demonstrated the effectiveness of a simple extension to kernel-based methods that utilizes empirical predictions rather than gradient kernels. This modification enables black-box batch active learning with good performance. Importantly, our work also generalizes to active learning with non-differentiable models, an area that has received limited attention as of late.

Our results also partially answer one of the research questions posed by Kirsch & Gal (2022): how do prediction-based methods compare to parameter-based ones? We find that for regression the prediction-based methods are competitive with the parameter-based methods in batch active learning.

The main limitation of our proposed approach lies in the acquisition of a sufficient amount of empirical predictions. This could be a challenge, particularly when using deep ensembles with larger models or non-differentiable models that cannot be parallelized efficiently. Our experiments using virtual ensembles indicate that the diversity of the ensemble members plays a crucial role in determining the performance.

References

Christos Anagnostopoulos, Fotis Savva, and Peter Triantafillou. Scalable aggregation predictive analytics. *Applied Intelligence*, 48(9):2546–2567, 2018.

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

Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham Kakade. Gone fishing: Neural active learning with fisher embeddings, 2021.

Rafael Ballester-Ripoll, Enrique G Paredes, and Renato Pajarola. Sobol tensor trains for global sensitivity analysis. *Reliability Engineering & System Safety*, 183:311–322, 2019.

Freddie Bickford Smith, Andreas Kirsch, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-Oriented Bayesian Active Learning. *AI Stats*, 2023.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Robert Burbidge, Jem J Rowland, and Ross D King. Active learning for regression based on query by committee. *Lecture Notes in Computer Science*, 4881:209–218, 2007.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15:201–221, 1994.

Tewodors Deneke, Habtegebreil Haile, Sébastien Lafond, and Johan Lilius. Video transcoding time prediction for proactive load balancing. In *International Conference on Multimedia and Expo*, 2014.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 1972.

Jerome H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pp. 1–67, 1991.

Yarin Gal and Z Ghahramani. Dropout as a bayesian approximation: representing model uncertainty in deep learning. arxiv. *arXiv preprint arXiv:1506.02142*, 2015.

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

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

Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2D image registration in ct images using radial image descriptors. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2011.

David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart. A framework and benchmark for deep batch active learning for regression. *arXiv preprint arXiv:2203.09410*, 2022.

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

Alexander Immer. Disentangling the gauss-newton method and approximate inference for neural networks. *arXiv preprint arXiv:2007.11994*, 2020.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, pp. 703–711. PMLR, 2021.

Andrew Jesson, Panagiotis Tigas, Joost van Amersfoort, Andreas Kirsch, Uri Shalit, and Yarin Gal. Causalbald: Deep bayesian active learning of outcomes to infer treatment-effects from observational data. *Advances in Neural Information Processing Systems*, 34:30465–30478, 2021.

Seho Kee, Enrique Del Castillo, and George Runger. Query-by-committee improvement with diversity and density in batch active learning. *Information Sciences*, 454:401–418, 2018.

Andreas Kirsch and Yarin Gal. Unifying approaches in active learning and active sampling via fisher information and information-theoretic quantities. *Transactions on Machine Learning Research*, 2022.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32, 2019.

Andreas Kirsch, Sebastian Farquhar, Parmida Atighehchian, Andrew Jesson, Frederic Branchaud-Charron, and Yarin Gal. Stochastic batch acquisition for deep active learning. *CoRR*, 2106:4, 2021.

Suraj Kothawade, Vishal Kaushal, Ganesh Ramakrishnan, Jeff Bilmes, and Rishabh Iyer. Prism: A rich class of parameterized submodular information measures for guided data subset selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10238–10246, 2022.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7, 1994.

Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1193–1200, 2011.

B Lakshminarayanan, A Pritzel, and C Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. arxiv. *arXiv preprint arXiv:1612.01474*, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Miguel Lázaro-Gredilla and Aníbal R Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *IEEE transactions on neural networks*, 21(8):1345–1351, 2010.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pp. 986–1005, 1956.

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4): 590–604, 1992.

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.

Mehdi Neshat, Bradley Alexander, Markus Wagner, and Yuanzhong Xia. A detailed comparison of meta-heuristic methods for optimising wave energy converter placements. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018.

Hieu T Nguyen, Joseph Yadegar, Bailey Kong, and Hai Wei. Efficient batch-mode active learning of random forest. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 596–599. IEEE, 2012.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. *Advances in neural information processing systems*, 32, 2019.

L Prokhorenkova, G Gusev, A Vorobev, AV Dorogush, and A Gulin. Catboost: Unbiased boosting with categorical features. arxiv 2017. *arXiv preprint arXiv:1706.09516*, 2017.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Tim G. J. Rudner, Zonghao Chen, Yee Whye Teh, and Yarin Gal. Tractabe Function-Space Variational Inference in Bayesian Neural Networks. 2022.

Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou. Explaining aggregates for exploratory analytics. In *International Conference on Big Data*, 2018.

Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

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 literature survey. 2009.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 287–294, 1992.

Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

Dominik Ślęzak, Marek Grzegorowski, Andrzej Janusz, Michał Kozielski, Sinh Hoa Nguyen, Marek Sikora, Sebastian Stawicki, and Łukasz Wróbel. A framework for learning and embedding multi-sensor forecasting models into a decision support system: A case study of methane concentration in coal mines. *Information Sciences*, 451:112–133, 2018.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Xueying Zhan, Qingzhong Wang, Kuan-hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B Chan. A comparative survey of deep active learning. *arXiv preprint arXiv:2203.13450*, 2022.

A Experiment Details

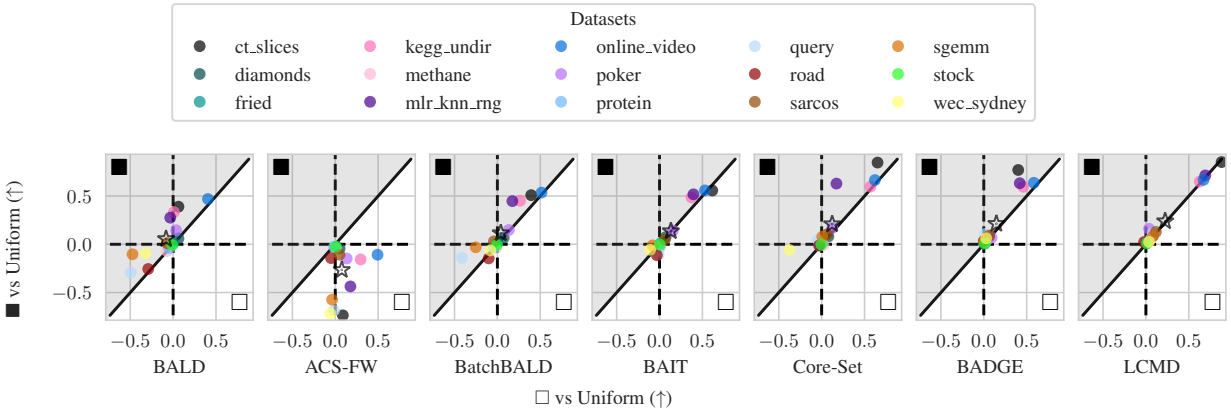


Figure 3: *Final Logarithmic RMSE by regression datasets for DNNs:* ■ *vs* □ *(vs Uniform).* Across acquisition functions, the performance of black-box methods is highly correlated with the performance of white-box methods, even though black-box methods make fewer assumptions about the model. We plot the improvement of the white-box method (□) over the uniform baseline on the x-axis, so for datasets with markers left of the dashed vertical lines, the white-box method performs better than uniform, and the improvement of the black-box method (■) over the uniform baseline on the y-axis, so for datasets with markers above the dashed horizontal lines, the black-box method performs better than uniform. Similarly, for datasets with markers in the ■ region, the black-box method performs better than the white-box method. The average over all datasets is marked with a star $\star$.

Table 1: *Average performance of black-box* ■ *and white-box* □ *batch active learning acquisition functions using DNNs.* On average, for five acquisition methods, the black-box method performs better than the white-box method. Cf. Figure 2, which analyzes the final epoch.

Acquisition function	MAE	RMSE	95%	99%	MAXE
Uniform	-1.934	-1.401	-0.766	-0.163	1.107
■ **BALD**	-1.794	-1.389	-0.713	-0.221	0.946
□ BALD	-1.722	-1.285	-0.614	-0.077	1.080
■ **BatchBALD**	-1.865	-1.465	-0.792	-0.303	0.892
□ BatchBALD	-1.895	-1.463	-0.808	-0.288	0.916
□ BAIT	-1.998	-1.541	-0.895	-0.357	0.888
■ **BAIT**	-1.892	-1.489	-0.817	-0.328	0.881
□ ACS-FW	-1.937	-1.439	-0.793	-0.225	1.016
■ **ACS-FW**	-1.678	-1.168	-0.509	0.085	1.278
■ **Core-Set**	-1.988	-1.585	-0.926	-0.435	**0.831**
□ Core-Set	-1.923	-1.490	-0.831	-0.307	0.929
■ **BADGE**	-2.042	-1.579	-0.931	-0.383	0.948
□ BADGE	-2.007	-1.530	-0.895	-0.329	1.008
■ **LCMD**	**-2.048**	**-1.609**	**-0.965**	**-0.437**	0.874
□ LCMD	-2.033	-1.589	-0.940	-0.402	0.914

Table 2: *Average performance of black-box* ■ *batch active learning acquisition functions on non-differentiable models.*

(a) Random Forests

Acquisition function	MAE	RMSE	95%	99%	MAXE
Uniform	-1.516	-0.975	-0.293	0.290	1.376
■ **BALD**	-1.222	-0.815	-0.106	0.365	1.348
■ **BatchBALD**	-1.449	-1.070	-0.401	0.064	**1.195**
■ **BAIT**	-1.582	**-1.158**	**-0.485**	**0.021**	1.198
■ **ACS-FW**	-1.502	-0.989	-0.310	0.266	1.379
■ **Core-Set**	-1.449	-1.071	-0.403	0.059	1.196
■ **BADGE**	**-1.618**	-1.150	-0.480	0.070	1.273
■ **LCMD**	-1.564	-1.116	-0.450	0.097	1.270

(b) Gradient-Boosted Decision Trees

Acquisition function	MAE	RMSE	95%	99%	MAXE
Uniform	-1.675	-1.172	-0.533	0.068	1.232
■ **BALD**	-1.353	-0.979	-0.308	0.182	1.225
■ **BatchBALD**	-1.474	-1.103	-0.448	0.052	1.134
■ **BAIT**	-1.497	-1.122	-0.467	0.034	1.137
■ **ACS-FW**	-1.501	-1.000	-0.354	0.241	1.346
■ **Core-Set**	-1.573	-1.193	-0.552	-0.036	**1.100**
■ **BADGE**	**-1.709**	**-1.259**	**-0.621**	-0.050	1.156
■ **LCMD**	-1.688	-1.256	-0.616	**-0.061**	1.132

Table 3: *Overview over used data sets.* See (Ballester-Ripoll et al., 2019; Neshat et al., 2018; Graf et al., 2011; Shannon et al., 2003; Deneke et al., 2014; Anagnostopoulos et al., 2018; Savva et al., 2018; Friedman, 1991; Ślęzak et al., 2018). The second column entries are hyperlinks to the respective web pages. Taken from Holzmüller et al. (2022).

Short name	Initial pool set size	Test set size	Number of features	Source	OpenML ID	Full name
sgemm	192000	48320	14	UCI		SGEMM GPU kernel performance
wec_sydney	56320	14400	48	UCI		Wave Energy Converters
ct_slices	41520	10700	379	UCI		Relative location of CT slices on axial axis
kegg_undir	50407	12921	27	UCI		KEGG Metabolic Reaction Network (Undirected)
online_video	53748	13756	26	UCI		Online Video Characteristics and Transcoding Time
query	158720	40000	4	UCI		Query Analytics Workloads
poker	198720	300000	95	UCI		Poker Hand
road	198720	234874	2	UCI		3D Road Network (North Jutland, Denmark)
mlr_knn_rng	88123	22350	132	OpenML	42454	mlr_knn_rng
fried	31335	8153	10	OpenML	564	fried
diamonds	41872	10788	29	OpenML	42225	diamonds
methane	198720	300000	33	OpenML	42701	Methane
stock	45960	11809	9	OpenML	1200	BNG(stock)
protein	35304	9146	9	OpenML	42903	physicochemical-protein
sarcos	34308	8896	21	GPML		SARCOS data