Black-Box Batch Active Learning for Regression

Andreas Kirsch
OATML, Department of Computer Science
University of Oxford

Reviewed on OpenReview: https://openreview.net/forum?id=fvEvDlKko6

Abstract

Batch active learning is a popular approach for efficiently training machine learning models on large, initially unlabelled datasets by repeatedly acquiring labels for batches of data points. However, many recent batch active learning methods are white-box approaches and are often 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. Crucially, our method only relies on model predictions. 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. 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, using first or second-order derivatives (e.g. model embeddings)\textsuperscript{1}.

This can present a limitation in real-world scenarios where model internals or gradients might be expensive to access—or might not be accessible at all. This is particularly of concern for ‘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 API. 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 examine black-box batch active learning (B\textsuperscript{3}AL)\textsuperscript{2} for regression which is compatible with a wider range of machine learning models. By black-box, we mean that our approach only relies on model predictions and does not require access to model internals or gradients. 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 (Kothawade et al., 2021; 2022; Ash et al., 2019; 2021) 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

\textsuperscript{1}Model embeddings can also be seen as first-order derivatives of the score under regression in regard to the last layer.

\textsuperscript{2}The code is available at https://github.com/BlackHC/2302.08981.
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 as random forests or gradient-boosted trees. 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. Specifically, we evaluate black-box batch active learning on a diverse set of regression datasets. We focus on regression because classification using the same approach would require Laplace approximations, Monte Carlo sampling, or expectation propagation (Williams & Rasmussen, 2006; Hernández-Lobato et al., 2011). This would complicate a fair comparison between black-box and white-box approaches. For the same reason, we also do not consider proxy-based active learning methods (Coleman et al., 2019), which constitutes an orthogonal direction to our investigation. 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 find that we can already obtain excellent results with a small number of them. Our results demonstrate the label efficiency of B³AL for various machine learning models. For deep learning models, B³AL even performs better than the corresponding state-of-the-art white-box methods in many cases.

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 B³AL. In §4, we detail the experimental setup and provide the results of our experimental evaluation. Finally, §5 concludes with a discussion and directions for future research.

2 Related Work

Active learning has a rich history, 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 set of unlabeled data and acquiring labels for batches of points rather than individual points at a time. 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 (Kirsch & Gal, 2022). 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. 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 only 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 set 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 regressing causal treatment effects and adopt a sampling distribution based on individual-acquisition scores, a heuristic proposed 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 a kernel-based formulation. This trivially 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 derivation, we make use of a Bayesian model in the narrow sense that we require some stochastic parameters $\Omega$—the model parameters or bootstrapped training data in the case of non-differentiable models like random forests, for example—with a distribution\(^3\) $p(\omega)$ (Bickford Smith et al., 2023):

$$p(y, \omega | x) = p(y | x, \omega) p(\omega).$$

Bayesian model averaging (BMA) is performed by marginalizing over $p(\omega)$ to obtain the predictive distribution $p(y | x)$. We can use Bayesian inference to obtain a posterior $p(\omega | D)$ for additional data $D$ via:

$$p(\omega | D) \propto p(D | \omega) p(\omega),$$

where $p(D | \omega)$ is the likelihood of the data given the parameters $\Omega$ and $p(\omega | D)$ is the (new) posterior distribution over $\Omega$. Importantly, the choice of $p(\omega)$ 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 $D_{pool} = \{x_i^{pool}\}$ and a small initially labeled training set $D_{train} = \{(x_i^{train}, y_i^{train})\}$, or $D_{train} = \emptyset$. In the batch acquisition setting, we want to repeatedly acquire labels for a subset $\{x_i^{acq}\}$ of the pool set of a given acquisition batch size $B$ and add them to the training set $D_{train}$. 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 $D_{train}$. Once we have chosen such an acquisition batch $\{x_i^{acq}\}$ of unlabeled data, we acquire labels $\{y_i^{acq}\}$ for these samples and train a new model on the combined training set $D_{train} \cup \{x_i^{acq}, y_i^{acq}\}$ and repeat the process. Crucial to the success of active learning is the choice of acquisition function $A(\{x_i^{acq}\}; p(\omega))$ which is a function of the acquisition batch $\{x_i^{acq}\}$ and the distribution $p(\omega)$ 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:

$$Y | x, \omega \sim \mathcal{N}(\mu(x; \omega), \sigma^2_N).$$

\(^3\)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 $p(\omega)$. 

---

Published in Transactions on Machine Learning Research (7/2022)
Equivalently, $Y \mid x, \omega \sim \mu(x; \omega) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma_N^2)$. As usual, we assume that the noise is independent for different inputs $x$ and parameters $\omega$. Homoscedastic noise is a special case of the general heteroscedastic setting: the noise variance is simply a constant. Our approach can be extended to heteroscedastic noise by substituting a function $\sigma_N(x; \omega)$ for $\sigma_N$, but for this work we limit ourselves to the simplest case.

### 3.1 Kernel-based Methods

We build on Holzmüller et al. (2022) which expresses contemporary batch active learning methods using kernel-based 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.

**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 from input space to 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:

$$Y_1, \ldots, Y_n \mid x_1, \ldots, x_n \sim \mathcal{N}(0, \text{Cov}[\mu(x_1), \ldots, \mu(x_n)] + \sigma_N^2 I),$$

where $\mu(x)$ are the observation-noise free predictions as random variables and $\text{Cov}[\mu(x_1), \ldots, \mu(x_n)]$ is the covariance matrix of the predictions. The covariance matrix is defined via the kernel function $k(x, x')$:

$$\text{Cov}[\mu(x_1), \ldots, \mu(x_n)] = \left[k(x_i, x_j)\right]_{i,j=1}^{n,n}.$$  

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

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

$$k_{\text{grad}}(x; x' \mid \omega^*) \triangleq \nabla_{\omega} \mu(x; \omega^*) \nabla^2_{\omega} \left[-\log p(\omega^*)\right]^{-1} \nabla_{\omega} \mu(x'; \omega^*)^\top$$

is the canonical choice, where $\omega^*$ is a maximum likelihood or maximum a posteriori estimate (MLE, MAP) and $\nabla^2_{\omega} \left[-\log p(\omega^*)\right]$ is the Hessian of the negative log likelihood at $\omega^*$. Note that $\nabla_{\omega} \mu(x; \omega^*)$ is a row vector. Commonly, the prior is a Gaussian distribution with an identity covariance matrix, and thus $\nabla^2_{\omega} \left[-\log p(\omega^*)\right] = I$. The significance of this kernel lies in its relationship with the Fisher information matrix at $\omega^*$ (Immer, 2020; Immer et al., 2021; Kirsch & Gal, 2022), or equivalently, with the linearization of the loss function around $\omega^*$ (Holzmüller et al., 2022). This leads to a Gaussian approximation, which results in a Gaussian predictive posterior distribution when combined with a Gaussian likelihood. The use of the finite-dimensional gradient kernel thus 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 distributions 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 $x_1, \ldots, x_n$ as:

$$k_{\text{grad} \rightarrow \text{post}}(x_1, \ldots, x_n)(x; x' \mid \omega^*) \triangleq \nabla_{\omega} \mu(x; \omega^*) \left(\sigma_N^{-2} \begin{pmatrix} \nabla_{\omega} \mu(x_1; \omega^*) \\ \vdots \\ \nabla_{\omega} \mu(x_n; \omega^*) \end{pmatrix} \begin{pmatrix} \nabla_{\omega} \mu(x_1; \omega^*) \\ \vdots \\ \nabla_{\omega} \mu(x_n; \omega^*) \end{pmatrix}^\top \right)^{-1} \nabla_{\omega} \mu(x'; \omega^*)^\top.$$
The factor $\sigma_N^2$ originates from implicitly conditioning on $Y_i | x_i$, 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:

$$H[Y_1, \ldots, Y_n | x_1, \ldots, x_n] = \frac{1}{2} \log \det(Cov[\mu(x_1), \ldots, \mu(x_n)]) + C_n,$$

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. Please see §A in the appendix for more details on information-theoretic active learning.

### 3.2 Black-Box Batch Active Learning

We more formally introduce the empirical covariance kernel and compare it to the gradient kernel commonly used for centered predictions.

This is also simply known as the covariance kernel

$$K(x_i, x_j) \triangleq \text{Cov}_\Omega[\mu_{x_i}^\omega; \mu_{x_j}^\omega].$$

Compared to §3.1, we do not define the covariance via the kernel but the kernel via the covariance.

We define the predictive covariance kernel $k_{\text{pred}}(x_i, x_j)$ as the covariance of the predicted means:

$$k_{\text{pred}}(x_i; x_j) \triangleq \text{Cov}_\Omega[\mu_{x_i}^\omega; \mu_{x_j}^\omega] = \text{Cov}_\Omega[\mu_{x_i}^\omega - \frac{1}{K} \sum_{l=1}^{K} \mu_{x_i}^{\omega_l}; \mu_{x_j}^\omega - \frac{1}{K} \sum_{l=1}^{K} \mu_{x_j}^{\omega_l}],$$

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

This is also simply known as the covariance kernel in the literature (Shawe-Taylor et al., 2004). We use the prefix predictive to make clear that we look at the covariance of the predictions. The resulting Gram matrix is equal the covariance matrix of the predictions and positive definite (for positive $\sigma_N$ and otherwise positive semi-definite) and thus a valid kernel.

### 3.2.2 Empirical Predictive Covariance Kernel

For $K$ sampled model parameters $\omega_1, \ldots, \omega_K \sim p(\omega)$—for example, the members of a deep ensemble—the empirical predictive covariance kernel $k_{\text{pred}}(x_i; x_j)$ is the empirical estimate:

$$k_{\text{pred}}(x_i; x_j) \triangleq \frac{1}{K} \sum_{k=1}^{K} \mu_{x_i}^{\omega_k} \frac{1}{K} \sum_{l=1}^{K} \mu_{x_j}^{\omega_l} - \frac{1}{K^2} \sum_{l=1}^{K} \mu_{x_j}^{\omega_l},$$

with centered predictions $\bar{\mu}_{x_i}^{\omega_k} \triangleq \mu_{x_i}^{\omega_k} - \frac{1}{K} \sum_{l=1}^{K} \mu_{x_l}^{\omega_l}$. As we can write this kernel as an inner product, it also immediately follows that the empirical predictive covariance kernel is a valid kernel and positive semi-definite.
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 (predictive) covariance kernel. This section explicitly conditions on $D^{\text{train}}$. The result is simple but instructive:

**Proposition 3.1.** The posterior gradient kernel $k_{\text{grad} \rightarrow \text{post}(D^{\text{train}})}(x_i; x_j | \omega^*)$ is an approximation of the predictive covariance kernel $k_{\text{pred}}(x_i; x_j)$.

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

$$
\mu(x; \omega) \approx \mu(x; \omega^*) + \nabla_\omega \mu(x; \omega^*) \left( \omega - \omega^* \right).
$$

Choose $\omega^* = E_{\omega \sim p(\omega | D^{\text{train}})}[\omega]$ (BMA). Then we have $E_{\omega \sim p(\omega | D^{\text{train}})}[\mu(x; \omega)] \approx \mu(x; \omega^*)$. Overall this yields:

$$
k_{\text{pred}}(x_i; x_j) = \text{Cov}_{\omega \sim p(\omega | D^{\text{train}})}[\mu(x_i; \omega); \mu(x_j; \omega)]
\approx E_{\omega^* + \Delta \omega \sim p(\omega | D^{\text{train}})}[\nabla_\omega \mu(x_i; \omega^*) \Delta \omega, \nabla_\omega \mu(x_j; \omega^*) \Delta \omega] + \nabla_\omega \mu(x_i; \omega^*) E_{\omega \sim p(\omega | D^{\text{train}})}[\Delta \omega \Delta \omega^\top] \nabla_\omega \mu(x_j; \omega^*)
= \nabla_\omega \mu(x_i; \omega^*) \text{Cov}[\Omega | D^{\text{train}}] \nabla_\omega \mu(x_j; \omega^*)^\top
\approx k_{\text{grad} \rightarrow \text{post}(D^{\text{train}})}(x_i; x_j | \omega^*).
$$

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

$$
\text{Cov}[\Omega | D^{\text{train}}]^{-1} \approx \nabla^2_\omega[-\log p(\omega^* | D^{\text{train}})]
\approx \nabla^2_\omega[-\log p(D^{\text{train}} | \omega^*) - \log p(\omega^*)]
\approx \sigma^2_N \sum_i \nabla_\omega \mu(x^{\text{train}}_i; \omega^*)^\top \nabla_\omega \mu(x^{\text{train}}_i; \omega^*) - \nabla^2_\omega \log p(\omega^*),
$$

where we have used Bayes' theorem and that $p(D^{\text{train}})$ vanishes under differentiation—it is constant in $\omega$. 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. $\nabla^2_\omega[-\log p(\omega^*)]$ is the prior term. This matches (8). \(\square\)

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 in this case, too. With $\Omega \triangleq (\omega_1, \ldots, \omega_K)$ fixed—e.g. these could be the different parameters of the members of an ensemble—we introduce a latent $\Psi$ to represent the index of the ‘true’ hypothesis $\omega^* \in \Omega$ from this empirical hypothesis space $\hat{\Omega}$, which we want to identify. This is similar to QbC (Seung et al., 1992). In essence, the latent $\Psi$ takes on the role of $\Omega$ from the previous section, and we are interested in learning the ‘true’ $\Psi$ from additional data. We, thus, examine the kernels for $\Psi$, as opposed to $\Omega$.

Specifically, we model $\Psi$ using a one-hot categorical distribution, that is a multinomial distribution from which we draw one sample: $\Psi \sim \text{Multinomial}(q, 1)$, with $q \in S^{K-1}$ parameterizing the distribution, where $S^{K-1}$ denotes the $K - 1$ simplex in $\mathbb{R}^K$. Then, $q_k = p(\Psi = e_k)$, where $e_k$ denotes the $k$-th unit vector; and $\sum_{k=1}^{K} q_k = 1$. For the corresponding predictive mean $\hat{\mu}(x; \Psi)$, we have:

$$
\hat{\mu}(x; \Psi) \triangleq \mu(x; \omega_\Psi) = \langle \mu(x; \cdot), \Psi \rangle,
$$

where $\langle \cdot, \cdot \rangle$ is the dot product.
where we use \( \omega_{\psi} \) to denote the \( \omega_k \) when we have \( \psi = e_k \) in slight abuse of notation, and \( \mu(\mathbf{x}; \cdot) \in \mathbb{R}^K \) is a column vector of the predictions \( \mu(\mathbf{x}; \omega_k) \) for \( \mathbf{x} \) for all \( \omega_k \). This follows from \( \psi \) being a one-hot vector.

We now examine this model and its kernels. The BMA of \( \tilde{\mu}(\mathbf{x}; \Psi) \) matches the previous empirical mean if we choose \( q \) to have an uninformative\(^4\) uniform distribution over the hypotheses \( (q_k \overset{\Delta}{=} \frac{1}{K}) \):

\[
\tilde{\mu}(\mathbf{x}; q) \overset{\Delta}{=} \mathbb{E}_{p(\psi)}[\mu(\mathbf{x}; \omega_{\psi})] = \langle \mu(\mathbf{x}; \cdot), q \rangle = \sum_{\psi=1}^K q_\psi \mu(\mathbf{x}; \omega_{\psi}) = \frac{1}{K} \sum_{\psi=1}^K \mu(\mathbf{x}; \omega_{\psi}).
\]  

(25)

What is the predictive covariance kernel of this model? And what is the posterior gradient kernel for \( q \)?

**Proposition 3.2.**

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

2. The ‘posterior’ gradient kernel \( k_{\text{grad, } \psi \rightarrow \text{post}(\mathcal{D}^{\text{train}})}(\mathbf{x}_i; \mathbf{x}_j) \) for \( \hat{\Omega} \) in respect to \( \Psi \) using uniform \( q \) is equal to the empirical predictive covariance kernel \( k_{\text{pred}}(\mathbf{x}_i; \mathbf{x}_j) \).

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

\[
k_{\text{pred, } \psi}(\mathbf{x}_i; \mathbf{x}_j) = \text{Cov}_{\psi}[\tilde{\mu}(\mathbf{x}_i; \psi); \tilde{\mu}(\mathbf{x}_j; \psi)] = \mathbb{E}_{p(\psi)}[\tilde{\mu}(\mathbf{x}_i; \psi) \tilde{\mu}(\mathbf{x}_j; \psi)] \overset{(28)}{=} \frac{1}{K} \sum_{\psi} \mu(\mathbf{x}_i; \omega_{\psi}) \mu(\mathbf{x}_j; \omega_{\psi}) = k_{\text{pred}}(\mathbf{x}_i; \mathbf{x}_j).
\]  

(26)

For the second statement, we will show that we can express the predictive covariance kernel as a linearization around \( \Psi \). We can read off a linearization for \( \nabla_\psi \tilde{\mu}(\mathbf{x}_i; \psi) \) from the inner product in Equation (25):

\[
\nabla_\psi \tilde{\mu}(\mathbf{x}_i; \psi) = \mu(\mathbf{x}_i; \cdot)^T,
\]  

(27)

This allows us to write the predictive covariance kernel as a linearization around \( q \):

\[
k_{\text{pred, } \psi}(\mathbf{x}_i; \mathbf{x}_j) = \text{Cov}_{\psi \rightarrow p(\psi)}[\tilde{\mu}(\mathbf{x}_i; \psi); \tilde{\mu}(\mathbf{x}_j; \psi)] \overset{(28)}{=} \mathbb{E}_{q + \Delta \psi \rightarrow p(\psi)}[\nabla_\psi \tilde{\mu}(\mathbf{x}_i; \psi) \Delta \psi, \nabla_\psi \tilde{\mu}(\mathbf{x}_j; \psi) \Delta \psi] \overset{(29)}{=} \nabla_\psi \tilde{\mu}(\mathbf{x}_i; q) \text{ Cov}[\Psi] \nabla_\psi \tilde{\mu}(\mathbf{x}_i; q)^T \overset{(30)}{=} k_{\text{grad, } \psi \rightarrow \text{post}(\mathcal{D}^{\text{train}})}(\mathbf{x}_i; \mathbf{x}_j).
\]  

(31)

The above gradient kernel is only the posterior gradient kernel in the sense that we have sampled \( \omega_{\psi} \) from the non-differentiable model after inference on training data. The samples themselves are drawn uniformly.

The covariance of the multinomial \( \Psi \) is: \( \text{Cov}[\Psi] = \text{diag}(q) - qq^T \). Thus, substituting, we can verify that the posterior gradient kernel is indeed equal to the predictive covariance kernel explicitly once more:

\[
k_{\text{grad, } \psi \rightarrow \text{post}(\mathcal{D}^{\text{train}})}(\mathbf{x}_i; \mathbf{x}_j) = \nabla_\psi \tilde{\mu}(\mathbf{x}_i; q) (\text{diag}(q) - qq^T) \nabla_\psi \tilde{\mu}(\mathbf{x}_i; q)^T \overset{(32)}{=} \mu(\mathbf{x}_i; \cdot)^T \text{diag}(q) \mu(\mathbf{x}_j; \cdot) - (\mu(\mathbf{x}_i; \cdot)^T q (q^T \mu(\mathbf{x}_j; \cdot)))
\]  

(33)

\[
= \frac{1}{K} \sum_{\psi} \mu(\mathbf{x}_i; \omega_{\psi}) \mu(\mathbf{x}_j; \omega_{\psi})^T - \left( \frac{1}{K} \sum_{\psi} \mu(\mathbf{x}_i; \omega_{\psi}) \right) \left( \frac{1}{K} \sum_{\psi} \mu(\mathbf{x}_j; \omega_{\psi}) \right)
\]  

(34)

\[
= k_{\text{pred}}(\mathbf{x}_i; \mathbf{x}_j).
\]  

(35)

\(^4\)If we had additional information about the \( \Omega \)—for example, if we had validation losses—we could use that to inform \( q \).
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).
We follow the evaluation from Holzmüller et al. (2022) and use their framework 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). The kernels and selection methods used are detailed in Table 1 in the appendix. We hypothesize that the virtual ensembles and a bagged ensemble of random forests do not express as much predictive disagreement which leads to worse performance for active learning. In the appendix, see Table 3 for average performance metrics and Figure 8 and 9 for plots with additional error metrics. Averaged over at least 5 trials.

4 Results

We report the logarithmic RMSE averaged over 5 trials for each dataset and method. For ensembles,
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 right 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). For datasets with markers in the diagonal half, 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 slightly better than the white-box methods for all but ACS-FW and BAIT. In the appendix, see Figure 5 for the final acquisition round and Table 2 for details on the datasets. Averaged over at least 5 trials.

we compute the performance for each ensemble member separately, enabling a fair comparison to the non-ensemble methods. Performance differences can thus be attributed to the acquisition function, rather than the ensemble. We used A100 GPUs with 40GB of GPU memory.

**Ensemble Size.** For deep learning, we use a small ensemble of 10 models, which is sufficient to achieve good performance. This ensemble size can be considered ‘small’ in the regression setting (Lázaro-Gredilla & Figueiras-Vidal, 2010; Zhang & Zhou, 2013), whereas in Bayesian Optimal Experiment Design much higher sample counts are regularly used (Foster et al., 2021). In many cases, training an ensemble of regression models is still fast and could be considered cheap compared to the cost of acquiring additional labels. For non-differentiable models, we experiment with: a) random forests (Breiman, 2001), using the different trees as ensemble members; b) bagged random forests, each with 100 trees; and c) 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. We use the predictions from each tree as a virtual ensemble member. We do not perform any hyperparameter tuning. We also report results for random forests with bagging, where we train a real ensemble of 10 random forests. For gradient-boosted decision trees, we use a virtual ensemble of up to 20 members with early stopping using a validation set⁵. 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 3, we see that B³AL is competitive with white-box active learning, when using BALD, BatchBALD, BAIT, BADGE, and Core-Set. On average, B³AL 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 in the low data regime as the more explicit approximation in B³AL via ensembling.

---

⁵If 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.
Why Can Black-Box Methods Outperform White-Box Methods? Following §3, white-box and black-box methods are based on kernels, which can be seen as different approximations of the predictive covariance kernel. White-box methods implicitly assume that the predictive covariance kernel is well approximated by the Fisher information kernel and the gradient kernel (Kirsch & Gal, 2022). However, Long (2022) demonstrated that this assumption does not always hold, particularly in low data regimes, where a Gaussian might not approximate the parameter distribution well. Instead, Long (2022) suggests using a multimodal distribution. In these situations, methods that employ ensembling, such as B³AL, to approximate the predictive covariance kernel can be more robust. The different ensemble members can reside in different modes of the parameter distribution, allowing black-box methods to outperform their white-box counterparts.

Non-Differentiable Models. In Figure 1b, 2a, and 2b, we observe that B³AL is effective for non-differentiable models, including random forests and gradient-boosted decision trees. BALD for non-differentiable models can be considered equivalent to QbC (Seung et al., 1992), while BatchBALD for non-differentiable models can be viewed as equivalent to QbC with batch acquisition (Nguyen et al., 2012). For random forests, all methods except BALD (using top-k selection) outperform uniform acquisition. However, for random forests with bagging and gradient-boosted decision trees, B³AL surpasses random acquisition only when employing LMCD and BADGE. This may be attributed to the reduced disagreement within a virtual ensemble for gradient-boosted decision trees and between distinct random forests. In particular, random forests with bagging appear to support this explanation, as a single random forest seems to exhibit...
more disagreement among its individual trees than an ensemble of random forests with bagging does between different forests. This is evident in the superior overall active learning performance of the single random forest compared to the ensemble of random forests with bagging, c.f. Figure 1b and 2a.

4.1 Ablations

Here, we discuss two ablations of B³AL: the effect of the ensemble size and the effect of the acquisition batch sizes.

Acquisition Batch Size. The right-hand plots in Figures 1 and 2 show that the performance generally decreases with increasing acquisition batch size for all acquisition approaches. Comparing white-box and black-box methods in Figure 1a, we see that only at the largest batch size of 4096, the white-box methods perform as well as the black-box methods for neural networks.

Ensemble Batch Size. Figure 4 shows the effect of the ensemble size on the performance of B³AL for random forests. Increasing the ensemble size generally improves the performance for all acquisition approaches except LMCD, for which performance first improves and then degrades. The performance increase with uniform sampling is due to better model predictions. This shows that the improvements stemming from more informative sample acquisition by the active learning methods due to increasing ensemble sizes can be significantly larger than the improvements due to better model predictions.

Best-performing Kernels and Selection Methods. Holzmüller et al. (2022) evaluate different kernels and variants of selection methods based on the SotA methods. The best-performing kernels and selection methods are shown in Table 5 in the appendix. We run an ablation comparing black-box methods with the best-performing white-box kernels from Holzmüller et al. (2022). These variants do not actually match prior art—we use the same names for the acquisition functions to denote the origin of the variants. The results are shown in Appendix B.1. We observe that B³AL also performs on par or better than the best-performing white-box kernels.

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, B³AL also generalizes to non-differentiable models, an area that has received limited attention as of late.

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. The main limitation of the empirical comparisons is that we only consider regression tasks. Extending the results to classification is an important direction for future work.

Overall, our results 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.

Acknowledgements

The author would like to thank the anonymous TMLR reviewers for their patience and kind feedback during the review process. Their feedback has significantly improved this work. Likewise, many thanks to David Holzmüller for their constructive and helpful feedback, as well as for making their framework and results easily available. AK is supported by the UK EPSRC CDT in Autonomous Intelligent Machines and Systems (grant reference EP/L015897/1). ChatGPT was used to suggest specific text and table edits.
References


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


A Information-Theoretic Active Learning

The following section provides a brief overview of information-theoretic active learning, focusing on the expected information gain/BALD and greedy batch selection. Other information-theoretic quantities can be examined similarly. We refer to Kirsch & Gal (2022) for more details. For other selection methods, we refer to Holzmüller et al. (2022).

Expected Information Gain (EIG) (Lindley, 1956) is based on the mutual information. For a Bayesian model, the EIG between the parameters \( \Omega \) and the predictions \( Y | x \) is defined as:

\[
I[\Omega; Y | x] = H[\Omega] - H[\Omega | Y, x] = \int p(\omega, y | x) \log \frac{p(\omega | y, x)}{p(\omega)} \, d\omega \, dy,
\]

(36)

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

Computing the EIG can be difficult when focused on the model parameters, as we need to compute the conditional entropy \( H[\Omega | Y, x] \)—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 \( H[\Omega] \), as it does not depend on the data \( x \) 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:

\[
I[\Omega; Y | x] = I[Y; \Omega | x] = H[Y | x] - H[Y | \Omega, x] = \int p(\omega, y | x) \log \frac{p(y | x, \omega)}{p(y | x)} \, d\omega \, dy,
\]

(37)

This can be much easier to evaluate by sampling from \( \Omega \) without the need for additional Bayesian inference. BatchBALD (Kirsch et al., 2019) is an extension of BALD to the batch acquisition of samples \( \{x^{acq}_i\} \) for classification tasks:

\[
I[\Omega; Y_1, \ldots, Y_B | x_1, \ldots, x_B] = H[Y_1, \ldots, Y_B | x_1, \ldots, x_B] - \sum_i H[Y_i | x_i, \Omega],
\]

(38)

using the mutual information between the parameters \( \omega \) and the predictions \( \{Y_i\} \) on an acquisition candidate batch \( \{x_i\} \), 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 for the observation noise, yielding an upper-bound overall.

The same approach can be applied to other information quantities like the expected predicted information gain (Bickford Smith et al., 2023). 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 \( \mathcal{D}^{pool} \) that maximizes \( A(\{x_i\}, p(\omega)) \).

Finding the optimal solution is NP-hard (Schrijver et al., 2003). Kirsch et al. (2019) find that \( A_{\text{BatchBALD}} \) is submodular and that iteratively adding the sample \( x^{acq}_i \) with the highest \( A_{\text{BatchBALD}}(x^{acq}_1, \ldots, x^{acq}_i, p(\omega)) \) to the acquisition batch size is a \( 1 - \frac{1}{e} \)-approximation. This is a well-known result in submodular optimization and is known as the greedy algorithm (Nemhauser et al., 1978). For the entropy approximation, finding an optimal solution is equivalent to finding the mode of the k-DPP (Kulesza & Taskar, 2011).
Table 1: Kernels and selection methods for SotA methods. Taken from Holzmüller et al. (2022). See Holzmüller et al. (2022) for more details.

<table>
<thead>
<tr>
<th>Known as</th>
<th>Selection method</th>
<th>Kernel</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>BALD (Houlsby et al., 2011)</td>
<td>MaxDiag</td>
<td>$k_{il \rightarrow x_{train}}$</td>
<td>for GP with kernel $k$</td>
</tr>
<tr>
<td>BatchBALD (Kirsch et al., 2019)</td>
<td>MaxDet-P</td>
<td>$k_{il \rightarrow x_{train}}$</td>
<td>for GP with kernel $k$, proposed for classification</td>
</tr>
<tr>
<td>BAIT (Ash et al., 2021)</td>
<td>BAIT-FB-P</td>
<td>$k_{il \rightarrow x_{train}}$</td>
<td>proposed for classification</td>
</tr>
<tr>
<td>ACS-FW (Pinsler et al., 2019)</td>
<td>FrankWolFE-P</td>
<td>$k_{il \rightarrow x_{train}}$</td>
<td></td>
</tr>
<tr>
<td>Core-Set (Sener &amp; Savarese, 2017), FF-Active (Geifman &amp; El-Yaniv, 2017)</td>
<td>MaxDist-TP*</td>
<td>$k_{il}$</td>
<td>proposed for classification</td>
</tr>
<tr>
<td>BADGE (Ash et al., 2019)</td>
<td>KMeansPP-P</td>
<td>$k_{il \rightarrow x_{train}}$</td>
<td></td>
</tr>
<tr>
<td>LCMD</td>
<td>LCMD-TP</td>
<td>$k_{grad \rightarrow sketch(512)}$</td>
<td></td>
</tr>
</tbody>
</table>

* This refers to their simpler k-center-greedy selection method.

Table 2: Overview over used data sets. See Ballester-Ripoll et al. (2019); Neshat et al. (2018); Graf et al. (2011); Shannon et al. (2003); Denecke 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).

<table>
<thead>
<tr>
<th>Short name</th>
<th>Initial pool set size</th>
<th>Test set size</th>
<th>Number of features</th>
<th>Source</th>
<th>OpenML ID</th>
<th>Full name</th>
</tr>
</thead>
<tbody>
<tr>
<td>sgemm</td>
<td>192000</td>
<td>48320</td>
<td>14</td>
<td>UCI</td>
<td></td>
<td>SGEMM GPU kernel performance</td>
</tr>
<tr>
<td>wec_sydney</td>
<td>56320</td>
<td>14400</td>
<td>48</td>
<td>UCI</td>
<td></td>
<td>Wave Energy Converters</td>
</tr>
<tr>
<td>ct_slices</td>
<td>41520</td>
<td>10700</td>
<td>379</td>
<td>UCI</td>
<td></td>
<td>Relative location of CT slices on axial axis</td>
</tr>
<tr>
<td>kegg_mdir</td>
<td>50407</td>
<td>12921</td>
<td>27</td>
<td>UCI</td>
<td></td>
<td>KEGG Metabolic Reaction Network (Undirected)</td>
</tr>
<tr>
<td>online_video</td>
<td>53748</td>
<td>13756</td>
<td>26</td>
<td>UCI</td>
<td></td>
<td>Online Video Characteristics and Transcoding Time</td>
</tr>
<tr>
<td>query</td>
<td>158720</td>
<td>40000</td>
<td>4</td>
<td>UCI</td>
<td></td>
<td>Query Analytics Workloads</td>
</tr>
<tr>
<td>poker</td>
<td>198720</td>
<td>300000</td>
<td>95</td>
<td>UCI</td>
<td></td>
<td>Poker Hand</td>
</tr>
<tr>
<td>road</td>
<td>198720</td>
<td>234874</td>
<td>2</td>
<td>UCI</td>
<td></td>
<td>3D Road Network (North Jutland, Denmark)</td>
</tr>
<tr>
<td>mlr_knn_rng</td>
<td>88123</td>
<td>22350</td>
<td>132</td>
<td>OpenML</td>
<td>42454</td>
<td>mlr_knn_rng</td>
</tr>
<tr>
<td>fried</td>
<td>31335</td>
<td>8153</td>
<td>10</td>
<td>OpenML</td>
<td>564</td>
<td>fried</td>
</tr>
<tr>
<td>diamonds</td>
<td>41872</td>
<td>10788</td>
<td>29</td>
<td>OpenML</td>
<td>42225</td>
<td>diamonds</td>
</tr>
<tr>
<td>methane</td>
<td>198720</td>
<td>300000</td>
<td>33</td>
<td>OpenML</td>
<td>42701</td>
<td>Methane</td>
</tr>
<tr>
<td>stock</td>
<td>45900</td>
<td>11809</td>
<td>9</td>
<td>OpenML</td>
<td>1200</td>
<td>BNG(stock)</td>
</tr>
<tr>
<td>protein</td>
<td>35304</td>
<td>9146</td>
<td>9</td>
<td>OpenML</td>
<td>42903</td>
<td>physicochemical-protein</td>
</tr>
<tr>
<td>sarcos</td>
<td>34308</td>
<td>8896</td>
<td>21</td>
<td>GPML</td>
<td></td>
<td>SARCOS data</td>
</tr>
</tbody>
</table>
Figure 5: 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 ⋆.

Table 3: 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 3, which analyzes the final epoch.

<table>
<thead>
<tr>
<th>Acquisition function</th>
<th>MAE</th>
<th>RMSE</th>
<th>95%</th>
<th>99%</th>
<th>MAXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>-1.935</td>
<td>-1.401</td>
<td>-0.766</td>
<td>-0.163</td>
<td>1.107</td>
</tr>
<tr>
<td>■ BALD</td>
<td>-1.794</td>
<td>-1.389</td>
<td>-0.713</td>
<td>-0.221</td>
<td>0.946</td>
</tr>
<tr>
<td>□ BALD</td>
<td>-1.717</td>
<td>-1.278</td>
<td>-0.611</td>
<td>-0.070</td>
<td>1.090</td>
</tr>
<tr>
<td>■ BatchBALD</td>
<td>-1.865</td>
<td>-1.465</td>
<td>-0.792</td>
<td>-0.303</td>
<td>0.892</td>
</tr>
<tr>
<td>□ BatchBALD</td>
<td>-1.894</td>
<td>-1.462</td>
<td>-0.808</td>
<td>-0.287</td>
<td>0.912</td>
</tr>
<tr>
<td>□ BAIT</td>
<td>-1.999</td>
<td>-1.540</td>
<td>-0.895</td>
<td>-0.356</td>
<td>0.895</td>
</tr>
<tr>
<td>■ BAIT</td>
<td>-1.892</td>
<td>-1.489</td>
<td>-0.817</td>
<td>-0.328</td>
<td>0.881</td>
</tr>
<tr>
<td>□ ACS-FW</td>
<td>-1.935</td>
<td>-1.436</td>
<td>-0.793</td>
<td>-0.222</td>
<td>1.031</td>
</tr>
<tr>
<td>■ ACS-FW</td>
<td>-1.678</td>
<td>-1.168</td>
<td>-0.509</td>
<td>0.085</td>
<td>1.278</td>
</tr>
<tr>
<td>■ Core-Set</td>
<td>-1.988</td>
<td>-1.585</td>
<td>-0.926</td>
<td>-0.435</td>
<td>0.831</td>
</tr>
<tr>
<td>□ Core-Set</td>
<td>-1.924</td>
<td>-1.491</td>
<td>-0.831</td>
<td>-0.308</td>
<td>0.929</td>
</tr>
<tr>
<td>■ BADGE</td>
<td>-2.042</td>
<td>-1.579</td>
<td>-0.931</td>
<td>-0.383</td>
<td>0.948</td>
</tr>
<tr>
<td>□ BADGE</td>
<td>-2.009</td>
<td>-1.531</td>
<td>-0.897</td>
<td>-0.331</td>
<td>1.009</td>
</tr>
<tr>
<td>■ LCMD</td>
<td>-2.048</td>
<td>-1.609</td>
<td>-0.965</td>
<td>-0.437</td>
<td>0.874</td>
</tr>
<tr>
<td>□ LCMD</td>
<td>-2.033</td>
<td>-1.589</td>
<td>-0.940</td>
<td>-0.401</td>
<td>0.913</td>
</tr>
</tbody>
</table>
Table 4: Average performance of black-box batch active learning acquisition functions on non-differentiable models.

(a) Random Forests

<table>
<thead>
<tr>
<th>Acquisition function</th>
<th>MAE</th>
<th>RMSE</th>
<th>95%</th>
<th>99%</th>
<th>MAXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>-1.516</td>
<td>-0.977</td>
<td>-0.292</td>
<td>0.290</td>
<td>1.380</td>
</tr>
<tr>
<td>BALD</td>
<td>-1.222</td>
<td>-0.815</td>
<td>-0.106</td>
<td>0.365</td>
<td>1.348</td>
</tr>
<tr>
<td>BatchBALD</td>
<td>-1.449</td>
<td>-1.070</td>
<td>-0.401</td>
<td>0.064</td>
<td>1.195</td>
</tr>
<tr>
<td>BAIT</td>
<td>-1.582</td>
<td>-1.158</td>
<td>-0.485</td>
<td>0.021</td>
<td>1.198</td>
</tr>
<tr>
<td>ACS-FW</td>
<td>-1.502</td>
<td>-0.989</td>
<td>-0.310</td>
<td>0.266</td>
<td>1.379</td>
</tr>
<tr>
<td>Core-Set</td>
<td>-1.449</td>
<td>-1.071</td>
<td>-0.403</td>
<td>0.059</td>
<td>1.196</td>
</tr>
<tr>
<td>BADGE</td>
<td>-1.618</td>
<td>-1.150</td>
<td>-0.480</td>
<td>0.070</td>
<td>1.273</td>
</tr>
<tr>
<td>LCMD</td>
<td>-1.564</td>
<td>-1.116</td>
<td>-0.450</td>
<td>0.097</td>
<td>1.270</td>
</tr>
</tbody>
</table>

(b) Random Forests (Bagging)

<table>
<thead>
<tr>
<th>Acquisition function</th>
<th>MAE</th>
<th>RMSE</th>
<th>95%</th>
<th>99%</th>
<th>MAXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>-1.438</td>
<td>-0.931</td>
<td>-0.237</td>
<td>0.317</td>
<td>1.375</td>
</tr>
<tr>
<td>BALD</td>
<td>-1.154</td>
<td>-0.771</td>
<td>-0.070</td>
<td>0.386</td>
<td>1.349</td>
</tr>
<tr>
<td>BatchBALD</td>
<td>-1.283</td>
<td>-0.923</td>
<td>-0.245</td>
<td>0.205</td>
<td>1.263</td>
</tr>
<tr>
<td>BAIT</td>
<td>-1.304</td>
<td>-0.944</td>
<td>-0.269</td>
<td>0.184</td>
<td>1.254</td>
</tr>
<tr>
<td>ACS-FW</td>
<td>-1.206</td>
<td>-0.732</td>
<td>-0.019</td>
<td>0.503</td>
<td>1.471</td>
</tr>
<tr>
<td>Core-Set</td>
<td>-1.369</td>
<td>-1.030</td>
<td>-0.377</td>
<td>0.071</td>
<td>1.194</td>
</tr>
<tr>
<td>BADGE</td>
<td>-1.524</td>
<td>-1.089</td>
<td>-0.413</td>
<td>0.114</td>
<td>1.261</td>
</tr>
<tr>
<td>LCMD</td>
<td>-1.496</td>
<td>-1.099</td>
<td>-0.433</td>
<td>0.074</td>
<td>1.219</td>
</tr>
</tbody>
</table>

(c) Gradient-Boosted Decision Trees

<table>
<thead>
<tr>
<th>Acquisition function</th>
<th>MAE</th>
<th>RMSE</th>
<th>95%</th>
<th>99%</th>
<th>MAXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>-1.691</td>
<td>-1.179</td>
<td>-0.543</td>
<td>0.065</td>
<td>1.224</td>
</tr>
<tr>
<td>BALD</td>
<td>-1.445</td>
<td>-1.044</td>
<td>-0.379</td>
<td>0.143</td>
<td>1.217</td>
</tr>
<tr>
<td>BatchBALD</td>
<td>-1.484</td>
<td>-1.092</td>
<td>-0.435</td>
<td>0.080</td>
<td>1.167</td>
</tr>
<tr>
<td>BAIT</td>
<td>-1.485</td>
<td>-1.088</td>
<td>-0.433</td>
<td>0.095</td>
<td>1.191</td>
</tr>
<tr>
<td>ACS-FW</td>
<td>-1.498</td>
<td>-0.999</td>
<td>-0.340</td>
<td>0.246</td>
<td>1.326</td>
</tr>
<tr>
<td>Core-Set</td>
<td>-1.600</td>
<td>-1.215</td>
<td>-0.566</td>
<td>-0.053</td>
<td>1.086</td>
</tr>
<tr>
<td>BADGE</td>
<td>-1.728</td>
<td>-1.264</td>
<td>-0.629</td>
<td>-0.048</td>
<td>1.157</td>
</tr>
<tr>
<td>LCMD</td>
<td>-1.689</td>
<td>-1.253</td>
<td>-0.613</td>
<td>-0.055</td>
<td>1.132</td>
</tr>
</tbody>
</table>
Figure 6: DNNs: Error Metrics over 15 regression datasets. We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 5 trials.
Figure 7: Random Forests: Error Metrics over 15 regression datasets (cont’d). We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 5 trials.
Figure 8: Random forests (Bagging): Error Metrics (cont’d). We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 5 trials.
Figure 9: *Gradient-Boosted Trees with Virtual Ensemble: Error Metrics over 15 regression datasets (cont’d).* We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 5 trials.
### B.1 Comparison on Neural Networks with Best-Performing Kernels and Selection Methods

Table 5: Best-performing white-box kernels and selection methods based on SotA methods according to results from Holzmüller et al. (2022). Taken from Holzmüller et al. (2022).

<table>
<thead>
<tr>
<th>Based on</th>
<th>Selection method</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>BALD (Houlsby et al., 2011)</td>
<td>MAXDIAG</td>
<td>$k_{\text{grad}} \rightarrow \text{sketch}(512) \rightarrow \text{acs-rf}(512)$</td>
</tr>
<tr>
<td>BatchBALD (Kirsch et al., 2019)</td>
<td>MAXDET-P</td>
<td>$k_{\text{grad}} \rightarrow \text{sketch}(512) \rightarrow X_{\text{train}}$</td>
</tr>
<tr>
<td>BAIT (Ash et al., 2021)</td>
<td>BAIT-F-P</td>
<td>$k_{\text{grad}} \rightarrow \text{sketch}(512) \rightarrow X_{\text{train}}$</td>
</tr>
<tr>
<td>ACS-FW (Pinsler et al., 2019)</td>
<td>FRANKWOLFE-P</td>
<td>$k_{\text{grad}} \rightarrow \text{sketch}(512) \rightarrow \text{acs-rf-hyper}(512)$</td>
</tr>
<tr>
<td>Core-Set (Sener &amp; Savarese, 2017), FF-Active (Geifman &amp; El-Yaniv, 2017)</td>
<td>MAXDIST-P</td>
<td>$k_{\text{grad}} \rightarrow \text{sketch}(512) \rightarrow X_{\text{train}}$</td>
</tr>
<tr>
<td>BADGE (Ash et al., 2019)</td>
<td>KMEANSPP-P</td>
<td>$k_{\text{grad}} \rightarrow \text{sketch}(512) \rightarrow \text{acs-rf}(512)$</td>
</tr>
<tr>
<td>LCMD</td>
<td>LCMD-TP</td>
<td>$k_{\text{grad}} \rightarrow \text{sketch}(512)$</td>
</tr>
</tbody>
</table>

Table 6: Average performance of black-box ■ and white-box □ batch active learning acquisition functions using DNNs for Table 5. On average, for five acquisition methods, the black-box method performs better than the white-box method. Cf. Figure 11, which analyzes the final epoch.

<table>
<thead>
<tr>
<th>Acquisition function</th>
<th>MAE</th>
<th>RMSE</th>
<th>95%</th>
<th>99%</th>
<th>MAXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>-1.935</td>
<td>-1.401</td>
<td>-0.766</td>
<td>-0.163</td>
<td>1.107</td>
</tr>
<tr>
<td>■ BALD</td>
<td>-1.794</td>
<td>-1.389</td>
<td>-0.713</td>
<td>-0.221</td>
<td>0.946</td>
</tr>
<tr>
<td>□ BALD</td>
<td>-1.779</td>
<td>-1.372</td>
<td>-0.692</td>
<td>-0.191</td>
<td>0.979</td>
</tr>
<tr>
<td>□ BatchBALD</td>
<td>-1.915</td>
<td>-1.512</td>
<td>-0.844</td>
<td>-0.349</td>
<td>0.866</td>
</tr>
<tr>
<td>■ BatchBALD</td>
<td>-1.865</td>
<td>-1.465</td>
<td>-0.792</td>
<td>-0.303</td>
<td>0.892</td>
</tr>
<tr>
<td>□ BAIT</td>
<td>-2.014</td>
<td>-1.586</td>
<td>-0.928</td>
<td>-0.413</td>
<td>0.864</td>
</tr>
<tr>
<td>■ BAIT</td>
<td>-1.892</td>
<td>-1.489</td>
<td>-0.817</td>
<td>-0.328</td>
<td>0.881</td>
</tr>
<tr>
<td>□ ACS-FW</td>
<td>-1.979</td>
<td>-1.544</td>
<td>-0.892</td>
<td>-0.363</td>
<td>0.920</td>
</tr>
<tr>
<td>■ ACS-FW</td>
<td>-1.678</td>
<td>-1.168</td>
<td>-0.509</td>
<td>0.085</td>
<td>1.278</td>
</tr>
<tr>
<td>■ Core-Set</td>
<td>-1.988</td>
<td>-1.585</td>
<td>-0.926</td>
<td>-0.435</td>
<td>0.831</td>
</tr>
<tr>
<td>□ Core-Set</td>
<td>-1.916</td>
<td>-1.515</td>
<td>-0.845</td>
<td>-0.351</td>
<td>0.865</td>
</tr>
<tr>
<td>■ BADGE</td>
<td>-2.042</td>
<td>-1.579</td>
<td>-0.931</td>
<td>-0.383</td>
<td>0.948</td>
</tr>
<tr>
<td>□ BADGE</td>
<td>-2.009</td>
<td>-1.570</td>
<td>-0.915</td>
<td>-0.387</td>
<td>0.921</td>
</tr>
<tr>
<td>■ LCMD</td>
<td>-2.048</td>
<td>-1.609</td>
<td>-0.965</td>
<td>-0.437</td>
<td>0.874</td>
</tr>
<tr>
<td>□ LCMD</td>
<td>-2.033</td>
<td>-1.589</td>
<td>-0.940</td>
<td>-0.401</td>
<td>0.913</td>
</tr>
</tbody>
</table>
Figure 10: *Average Logarithmic RMSE by regression datasets for DNNs: ■ vs □ (vs Uniform).*

Figure 11: *Final Logarithmic RMSE by regression datasets for DNNs: ■ vs □ (vs Uniform).*

Figure 12: *Mean logarithmic RMSE over 15 regression datasets for Deep Neural Networks using the best-performing white-box kernels and selection methods from Table 5. Even for the best-performing white-box kernels and selection methods as determined by Holzmüller et al. (2022), the black-box approach outperforms the white-box approach overall.*
Figure 13: DNNs: Error Metrics over 15 regression datasets using the strongest white-box kernels from Table 5. We report mean absolute error (MAE), root mean squared error (RMSE), 95% and 99% quantiles, and the maximum error (MAXE). Averaged over 5 trials.