Efficient Bayesian Updates for Deep Active Learning via Laplace Approximations

Denis Huseljic (✉) Marek Herde Lukas Rauch Paul Hahn Zhixin Huang
Daniel Kottke Stephan Vogt Bernhard Sick

University of Kassel, Intelligent Embedded Systems, Kassel, Germany
dhuseljic@uni-kassel.de

Abstract. Deep active learning (AL) selects batches of instances for annotation to avoid retraining deep neural networks (DNNs) after each new label. Employing a naive top-b selection can result in a batch of redundant (similar) instances. To address this, various AL strategies employ clustering techniques that ensure diversity within a batch. We approach this issue by substituting the costly retraining with an efficient Bayesian update. Our proposed update represents a second-order optimization step using the Gaussian posterior from a last-layer Laplace approximation. Thereby, we achieve low computational complexity by computing the inverse Hessian in closed form. We demonstrate that in typical AL settings, our update closely approximates retraining while being considerably faster. Leveraging our update, we introduce a new framework for batch selection through sequential construction, updating the DNN after each label acquisition. Furthermore, we incorporate our update into a look-ahead selection strategy as a feasible upper baseline approximating optimal batch selection. Our results highlight the potential of efficient updates to advance deep AL research.

Keywords: Deep Active Learning · Batch Selection · Bayesian Updates

1 Introduction

Active Learning (AL) sequentially selects instances for annotation by human experts, aiming to maximize model performance while minimizing labeling efforts. When combined with deep neural networks (DNNs), AL typically selects instances in batches rather than one at a time. The reason for this is that retraining DNNs after each label acquisition is computationally expensive, and delay can lead to additional costs since annotators' time is valuable [18].

In a naive top-b batch selection, a batch of b instances with the highest scores is chosen based on an informativeness measure. However, when many similar instances are present, this approach can result in significant redundancy within the batch (similar instances have a similar score). Many selection strategies have been developed to replace this naive selection [15]. These strategies often employ clustering techniques to ensure diverse batches, ensuring that instances within a batch are dissimilar to one another [15]. While effective in reducing redundancy, clustering does not guarantee optimal selection due to its heuristic motivation.

Orthogonal to these strategies, we explore the concept of efficient "retraining" in deep AL. If retraining were computationally feasible, researchers could place greater emphasis on the development of theoretically sound informativeness measures instead of using heuristic clustering approaches to ensure diversity. Additionally, selection strategies that aim to maximize future performance–strategies that have been shown to be near-optimal in traditional AL [34]–could be made feasible with DNNs. Therefore, we examine the concept of updating DNNs through a single optimization step as a proxy for retraining and explore its potential to enhance the AL process.

To underscore the requirements of such an update, consider strategies designed to maximize future performance. Typically, these use a look-ahead to select instances that significantly change model predictions. Specifically, they examine how adding unlabeled instances to the labeled pool and retraining the model affects predictions [34]. However, with a large number of unlabeled instances and the costly retraining process of DNNs, this approach becomes infeasible in deep AL. Therefore, instead of retraining, a highly efficient update method is required. The only work to realize this with DNNs is by [39], which employ an ensemble of DNNs combined with Monte Carlo (MC) updates via Bayes' theorem. Although this update makes a look-ahead feasible, it remains suboptimal for several reasons: (i) the update requires an ensemble of DNNs, making the actual retraining time and memory demands inefficient; (ii) the update does not accurately reflect the performance of full retraining; and (iii) the updating process becomes inefficient with an increasing number of ensemble members.

In this article, we propose an efficient update method for DNNs in the context of AL for classification. Specifically, we transform an arbitrary DNN into a Bayesian neural network (BNN) by employing a last-layer Laplace approximation (LA) [8]. While the closed-form expression of the posterior allows us to leverage second-order optimization techniques, we ensure low computational complexity by computing the required inverse Hessian analytically. Unlike the MC-based update used in [39], our approach does not require an ensemble of DNNs, making it easily applicable and both memory- and training-efficient [8]. Additionally, as we utilize a single DNN, we can leverage pretrained foundation models [30], which are an essential part of modern AL strategies [15,14]. The resulting update is fast and closely matches the performance of full retraining. Extensive studies across different data modalities demonstrate that our updates outperform the typically employed MC-based ones [39] in terms of speed and performance. Furthermore, we examine the proposed update in two distinct AL scenarios:

1. **Enhancing Existing Strategies with Immediate Label Utilization:** We propose a simple framework to improve existing strategies by immediately making use of acquired labels through the proposed updates. Rather than selecting the top-b highest-scoring instances simultaneously, we iteratively select the highest-scoring instance b times *but* update the model between each selection. This simple strategy, which approximates single-instance AL during batch construction, performs surprisingly well, outperforming naive top-b selection as well as selection strategies that employ clustering.

2. **Optimal AL with Look-Ahead Selection:** We investigate the potential of our update with a look-ahead selection strategy in an optimal AL setting. Specifically, we approximate an optimal selection strategy that maximizes future performance. Instead of retraining, we ensure computational feasibility by employing our update. The resulting strategy outperforms all competitors, showcasing that currently employed selection strategies have much potential for improvement.

Summary of Contributions

Efficient DNN Update: We propose an efficient update method for DNNs that employs a Laplace approximation and second-order optimization techniques. We enable low computational complexity through closed-form computation of the inverse Hessian.
Comprehensive Evaluation: We perform an extensive evaluation across data modalities, demonstrating that our update outperforms MC-based updates in both speed and accuracy.
Immediate Label Utilization: We develop a simple framework that employs our update to immediately incorporate acquired labels, improving existing selection strategies by updating the model during batch construction.
Optimal AL with Look-Ahead: We study our update in an optimal AL setting, making a near-optimal selection strategy as an upper baseline computationally feasible.

2 Related Work

Pool-based deep AL strategies can be divided into three types. Uncertainty-based methods, such as margin sampling [3,37] and BALD [13], assume that instances with high predictive uncertainty are most informative. When selecting batches, these strategies pick the top-b highest-scoring instances, leading to redundancy. In contrast, diversity-based approaches like Core-Set [36] aim to select a diverse batch of instances by considering the feature representations of labeled and unlabeled data. In practice, *hybrid strategies*, a combination of both types, have been shown to work well. BatchBALD [20] extends BALD by reducing redundant information within a batch. Badge [1] selects instances with high gradient norms and ensures diversity by employing k-MEANS++ in the gradient space. Typiclust [15] replaces the notion of uncertainty with typicality and selects typical instances from clusters obtained through k-MEANS.

Look-ahead strategies [34] remain underexplored in deep AL. They aim to select instances expected to improve the model's performance the most by retraining for all possible candidate instances. In non-deep settings, such approaches have been shown to achieve near-optimal selection [34] while offering convergence guarantees [45]. However, adapting these strategies to deep AL is challenging due to the computational cost of retraining. To the best of our knowledge, BEMPS [39] is the only strategy to implement a look-ahead mechanism in deep AL. They employ deep ensembles and MC-based updates via Bayes' theorem

(see Section 3). while this update is computationally efficient, we show that its performance falls short compared to full model retraining.

Continual learning [9] updates models by only using new data, retaining knowledge from the previous one. Related techniques [17,33] use conventional first-order optimization methods, deriving regularization terms from an LA that penalizes large deviations from prior knowledge. Unlike our method, these approaches require training over multiple epochs for the regularization term to have an impact. Used as an update (i.e., single optimization step), these strategies simplify to a first-order gradient step. Additionally, they assume large amounts of new data per task (thousands of instances), whereas our update method is designed batch construction in AL, ranging from a single to hundreds of instances. For example, a typical benchmark is to extend a dataset with a task consisting of all instance-label pairs of a new class (ca. 5,000 in MNIST).

More closely related to our work is **online learning** [16], which aims to sequentially and efficiently update models from incoming data streams. Traditional approaches often focus on linear [46] or shallow [21] models with maximum-margin classification. However, applying online learning to DNNs remains difficult due to issues such as convergence, vanishing gradients, and large model sizes [35,43]. [35] proposed a method that modifies a DNN's architecture to facilitate updates. We argue that this is restrictive in state-of-the-art settings, given the increasing reliance on pretrained foundation models [10,30]. Recently, [19] proposed Bayesian online inference, which is also used in [39]. This method samples hypotheses (e.g., via MC-Dropout) from a BNN's posterior and weights them by the likelihood for new arriving data. However, the empirical results raise concerns about its feasibility in high-dimensional parameter spaces. We refer to these as MC-based updates.

BNNs [40] induce a prior distribution on parameters of a DNN and learn a posterior distribution given data. MC-Dropout [12] uses dropout to obtain a distribution over predictions. While it is simple to use, its inference is inefficient, and it provides suboptimal uncertainty estimates [31]. Deep ensembles [23] are known for their superior uncertainty estimates but are train and memory inefficient [31]. LAs [8] approximate the posterior as a Gaussian, with the MAP estimate as the mean and the inverse Hessian as the covariance. As computing this Hessian is expensive for large DNNs, LA is often used only in the last layer [8].

Finally, we consider related approaches focusing on the **efficiency of AL**. Prior work [6] shows that smaller more efficient proxy models can be used for AL selection with minimal performance loss. Building on this, [44] employed the last layer of a DNN as a proxy model in their benchmark. Other efforts to improve efficiency revolve around sub-sampling strategies with warm starts [7]. In contrast, rather than improving efficiency of the AL cycle, we update the DNN *while constructing the batch*, immediately incorporating label information that guides the construction process.

3 Fast Bayesian Updates for Deep Neural Networks

In this section, we first introduce the general concept of Bayesian updates together with the variant MC-based updates [19,39]. Afterward, we propose our novel method focusing on an efficient update of the Gaussian posterior distribution via last-layer LAs. For an introduction to LA, we refer to [8].

3.1 Bayesian Updates

We focus on classification problems with instance space $\mathcal{X}$ and label space $\mathcal{Y} = \{0, \dots, K-1\}$. The primary goal in our setting is to efficiently incorporate the information of new instance-label pairs $\mathcal{D}^{\oplus} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$ into a DNN trained on dataset $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$. Retraining the entire network on the extended dataset $\mathcal{D} \cup \mathcal{D}^{\oplus}$ results in high computational cost for a large dataset $\mathcal{D}$. Conversely, using the new data solely can cause catastrophic forgetting [33].

For this purpose, we employ BNNs with Bayesian updates as an efficient alternative to retraining. The main idea of BNNs is to estimate the posterior distribution $p(\boldsymbol{\omega}|\mathcal{D})$ over the parameters $\boldsymbol{\omega} \in \Omega$ given the observed training data $\mathcal{D}$ using Bayes' theorem. The obtained posterior distribution over the parameters can then be used to specify the predictive distribution over a new instance's class membership via marginalization:

$$p(y|\boldsymbol{x}, \mathcal{D}) = \mathbb{E}_{p(\boldsymbol{\omega}|\mathcal{D})}[p(y|\boldsymbol{x}, \boldsymbol{\omega})] = \int p(y|\boldsymbol{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\omega}. \tag{1}$$

Thereby, the likelihood $p(y|\boldsymbol{x}, \boldsymbol{\omega}) = [\mathrm{softmax}(f^{\boldsymbol{\omega}}(\boldsymbol{x}))]_y$ denotes the probabilistic output of a DNN with parameters $\boldsymbol{\omega}$, where $f^{\boldsymbol{\omega}} : \mathcal{X} \to \mathbb{R}^K$ is a function outputting class-wise logits.[1]

The formulation in equation 1 provides a theoretically sound way to obtain updated predictions. In particular, this is because the probabilistic outputs $p(y|\boldsymbol{x}, \boldsymbol{\omega})$ do not directly depend on the training data $\mathcal{D}$. Consequently, to obtain an updated predictive distribution, we do not need to update the parameters $\boldsymbol{\omega}$ directly but only the posterior distribution $p(\boldsymbol{\omega}|\mathcal{D})$. The updated posterior distribution $p(\boldsymbol{\omega}|\mathcal{D}, \mathcal{D}^{\oplus})$ is found through Bayes' theorem, where the current posterior distribution $p(\boldsymbol{\omega}|\mathcal{D})$ is considered the prior and multiplied with the likelihood $p(y|\boldsymbol{x}, \boldsymbol{\omega})$ per instance-label pair $(\boldsymbol{x}, y) \in \mathcal{D}^{\oplus}$. As instances in $\mathcal{D}$ and $\mathcal{D}^{\oplus}$ are assumed to be independently distributed, we can simplify the likelihood and reformulate the parameter distribution as follows[2]:

$$p(\boldsymbol{\omega}|\mathcal{D}^{\oplus}, \mathcal{D}) \propto p(\boldsymbol{\omega}|\mathcal{D}) p(\mathcal{D}^{\oplus}|\mathcal{D}, \boldsymbol{\omega}) \overset{\text{i.i.d.}}{=} p(\boldsymbol{\omega}|\mathcal{D}) p(\mathcal{D}^{\oplus}|\boldsymbol{\omega}) = p(\boldsymbol{\omega}|\mathcal{D}) \prod_{(\boldsymbol{x}, y) \in \mathcal{D}^{\oplus}} p(y|\boldsymbol{x}, \boldsymbol{\omega}). \tag{2}$$

We refer to equation 2 as the *Bayesian update*.

[1] We denote the i-th element of a vector $\boldsymbol{b}$ as $[\boldsymbol{b}]_i = b_i$.

[2] We denote $p(y_1, \dots, y_N \mid \boldsymbol{x}_1, \dots, \boldsymbol{x}_N, \boldsymbol{\omega})$ with $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ as $p(\mathcal{D}|\boldsymbol{\omega})$.

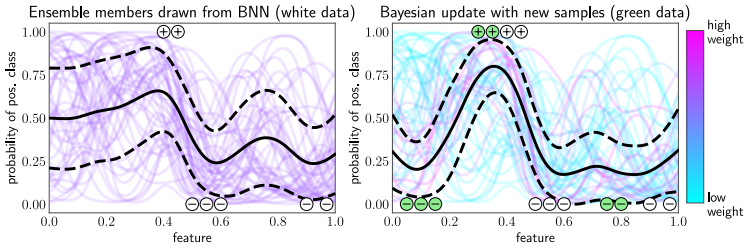

Fig. 1. The left plot shows the predicted probabilities of the positive class for each hypothesis (colored lines) drawn from a BNN as well as the mean (black solid line) and standard deviation (black dashed line) of its predictive distribution. The right plot shows updated weights for each hypothesis and the predictive distribution after observing additional instances (green).

The most common realization [19,39] of this update is through MC-based BNNs, such as MC-Dropout and deep ensembles. These BNNs rely on samples (or hypotheses) $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_M$ drawn from an approximate posterior $q(\boldsymbol{\omega}|\mathcal{D})$. Research [39,19] assumes that all hypotheses are equally likely to explain the observed data and have the same probability before updating. By updating the posterior distribution through equation 2, they weigh more likely hypotheses given the new data higher. We refer to these as MC-based updates with a formal definition given in Appendix A. Figure 1 illustrates this concept where different hypotheses $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_M \sim q(\boldsymbol{\omega}|\mathcal{D})$ are shown. Each hypothesis represents a possible true solution for the learning task (white instances). When new data (green instances) arrives, we weigh each hypothesis by its likelihood of explaining the new data and obtain an updated prediction without retraining. This results in an updated predictive distribution, as seen in bold in Figure 1 (right).

3.2 Fast Approximations of Bayesian Updates for Deep Neural Networks

Our update method is based on a combination of two concepts. First, instead of MC-based BNNs, we suggest using LAs on the last layer of a DNN. Second, we directly modify the approximate posterior distribution of the LA, providing a much more flexible way to adapt it to new data than reweighting. In the following, we explain each component in detail. For now, we focus on binary classification with $K = 2$, and refer to Appendix C for an extension to multi-class classification.

Last-layer LA: LAs approximate the (intractable) posterior distribution $p(\boldsymbol{\omega}|\mathcal{D})$ with a Gaussian centered on the maximum a posteriori (MAP) estimate with a covariance equal to the negative Hessian of the log posterior [8]. We denote this approximate distribution as

$$q(\boldsymbol{\omega}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\omega}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) \propto q(\boldsymbol{\omega}) \prod_{(\boldsymbol{x},y)\in\mathcal{D}} p(y|\boldsymbol{x}, \boldsymbol{\omega}), \tag{3}$$

where $q(\boldsymbol{\omega})$ is a Gaussian prior distribution. The MAP estimate $\hat{\boldsymbol{\mu}}$ results from training on $\mathcal{D}$ with conventional gradient optimization techniques. The covariance matrix $\hat{\boldsymbol{\Sigma}}$ is the inverse Hessian of the negative log posterior evaluated at the MAP estimate $\hat{\boldsymbol{\mu}}$ given training data $\mathcal{D}$. We model the posterior distribution only on the last layer of a DNN to ensure fast inference.

The benefits of using a last-layer LA are manifold. Given access to $q(\boldsymbol{\omega}|\mathcal{D})$ through a Gaussian, we enable *more flexible updates* compared to MC-based ones, as we can directly modify the mean and covariance. In contrast, MC-based updates only change the approximate distribution by reweighting hypotheses, leading to a strong dependency on the samples $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_M$. Last-layer LAs can be *integrated seamlessly* into nearly all DNNs, including pretrained models, as only the covariance has to be computed to obtain $q(\boldsymbol{\omega}|\mathcal{D})$. This is particularly important in deep AL, where recent findings highlight self-supervised learning as a crucial factor in selecting informative instances [15,14]. Finally, compared to deep ensembles and MC-Dropout, last-layer LAs introduce *minimal computational overhead*. While deep ensembles require longer training and MC-dropout impairs the inference time, LAs simply need to calculate a covariance matrix after training and allow fast inference through techniques such as mean-field approximation [27].

Second-Order Update: The second concept focuses on the update step of the Gaussian distribution. Observing new data, we follow the same approach as in equation 3, but with $q(\boldsymbol{\omega}|\mathcal{D})$ as our prior:

$$q(\boldsymbol{\omega}|\mathcal{D}, \mathcal{D}^{\oplus}) = \mathcal{N}(\boldsymbol{\omega}|\hat{\boldsymbol{\mu}}^{\text{upd}}, \hat{\boldsymbol{\Sigma}}^{\text{upd}}) \underset{\sim}{\propto} q(\boldsymbol{\omega}|\mathcal{D}) \prod_{(\boldsymbol{x},y)\in\mathcal{D}^{\oplus}} p(y|\boldsymbol{x}, \boldsymbol{\omega}), \tag{4}$$

where $\hat{\boldsymbol{\mu}}^{\text{upd}}$ and $\hat{\boldsymbol{\Sigma}}^{\text{upd}}$ represent the updated mean and covariance, respectively. The resulting updated posterior $q(\boldsymbol{\omega}|\mathcal{D}, \mathcal{D}^{\oplus})$ is non-Gaussian due to $p(y|\boldsymbol{x}, \boldsymbol{\omega})$ being a categorical likelihood. Consequently, the closed-form computation of the integral in equation 1 becomes intractable. The basic idea of our update is to approximate the new posterior $q(\boldsymbol{\omega}|\mathcal{D}, \mathcal{D}^{\oplus})$ by first applying a second-order optimization step via Gauss-Newton and then estimating the new covariance at that point. Thus, the updated mean and covariance are given by:

$$\hat{\boldsymbol{\mu}}^{\text{upd}} = \hat{\boldsymbol{\mu}} - \gamma \boldsymbol{H}^{-1}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}, \mathcal{D}^{\oplus}) \sum_{(\boldsymbol{x},y)\in\mathcal{D}^{\oplus}} (p_{\boldsymbol{x}} - y)\boldsymbol{h}_{\boldsymbol{x}}, \tag{5}$$

$$\hat{\boldsymbol{\Sigma}}^{\text{upd}} = \boldsymbol{H}^{-1}(\hat{\boldsymbol{\mu}}^{\text{upd}}, \hat{\boldsymbol{\Sigma}}, \mathcal{D}^{\oplus}), \tag{6}$$

where $\boldsymbol{h}_{\boldsymbol{x}}$ denotes the representation of $\boldsymbol{x}$ at the penultimate layer, $p_{\boldsymbol{x}} = \text{sigmoid}(\boldsymbol{h}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{\mu})$ is the probability for the positive class, and γ is a factor controlling the step size. The required updated Hessian can be computed efficiently in closed form following [38] by

$$\boldsymbol{H}^{-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{A}) = \boldsymbol{\Sigma} - \sum_{(\boldsymbol{x},y)\in\mathcal{A}} \frac{p_{\boldsymbol{x}}(1-p_{\boldsymbol{x}})}{1 + \overline{\sigma}_{\boldsymbol{x}} \cdot p_{\boldsymbol{x}}(1-p_{\boldsymbol{x}})} \big(\boldsymbol{\Sigma}\boldsymbol{h}_{\boldsymbol{x}}\big)\big(\boldsymbol{\Sigma}\boldsymbol{h}_{\boldsymbol{x}}\big)^{\mathrm{T}}, \tag{7}$$

where $\overline{\sigma}_{\boldsymbol{x}} = \boldsymbol{h}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{h}_{\boldsymbol{x}}$ is the predictive variance. The derivation can be found in Appendix D.

The idea behind using second-order optimization techniques is that they are more robust than first-order gradient optimization techniques due to the incorporation of curvature information of the log posterior. This results in a more accurate representation of the loss landscape, enabling more efficient and robust parameter updates that are less sensitive to hyperparameter choices. A critical aspect of our method's efficiency is that we do not need to recompute the Hessian from scratch. Instead, our updates leverage the covariance available through LAs and use the Woodbury identity [42] for closed-form inversion, significantly reducing computational overhead. Further, a common problem with last-layer LAs is that the Hessian can become a bottleneck when dealing with a large number of classes. To address this, we can approximate the Hessian in equation 7 by considering a Gaussian likelihood instead of a multi-class one, as also done in [25]. Lastly, we want to highlight that an assumption of an LA is that we are at the mode of a distribution, and adding more data violates this assumption. As we focus on AL and only update with a few (up to hundreds) instances at a time, this issue is less severe.

4 Bayesian Updating Experiments

In this section, we evaluate the efficiency of the proposed update by comparing it against competitors on various benchmark datasets for image and text classification. Our code is publicly available at `https://github.com/dhuseljic/dal-toolbox`.

4.1 Experimental Setup

Our **experimental design** is based on the work of [19]. First, we train a DNN on the training dataset $\mathcal{D}$ (baseline). We then use this baseline DNN to evaluate a last-layer LA and related Bayesian updates on additional instance-label pairs $\mathcal{D}^{\oplus}$ and compare these results to retraining the DNN on the complete dataset $\mathcal{D} \cup \mathcal{D}^{\oplus}$. We evaluate (i) the influence of the step size γ on chosen validation datasets, (ii) the impact of our update at different learning stages of the DNN, (iii) the impact of our update with increasing sizes of new arriving datasets, and (iv) the time efficiency of our update by considering the speed-up factor against retraining. For comparison, we consider MC-based updates by sampling 10k hypotheses from the approximate Gaussian posterior $q(\boldsymbol{\omega}|\mathcal{D})$ and the less complex first-order updates only considering gradients. Note that the latter is equivalent to the continual learning strategy of [33], as we demonstrate in Appendix A. Since first-order updates do not use the Hessian, this comparison also allows us to assess the benefits of using second-order optimization. We exclude retraining solely on $\mathcal{D}^{\oplus}$, as we empirically found that it leads to catastrophic forgetting [17]. All performance metrics are averaged across 10 repetitions. For visual clarity, we do not report standard errors.

The datasets $\mathcal{D}$ and $\mathcal{D}^{\oplus}$ are randomly sampled from real-world datasets. We use three image and three text **benchmark datasets** commonly used in

Table 1. Overview of datasets.

Type	Dataset	# classes
Image	Cifar-10 [22]	10
	Snacks [28]	20
	DTD [5]	47
Text	DBPedia [2]	14
	Banking-77 [4]	77
	Clinc-150 [24]	150

literature [15,32] with varying complexity reflected through different numbers of classes. Table 1 gives an overview. A detailed summary for each dataset is provided in Appendix E.

The goal of an update method is to ensure both effectiveness and speed. To assess this, we use different **performance metrics**. To evaluate *effectiveness*, or how well an update or retraining generalizes, we measure accuracy. When experimenting with hyperparameters, accuracy is assessed on a 10% validation split. Otherwise, it is measured on the test dataset. An optimal update method should achieve the same performance as completely retraining the DNN with $\mathcal{D} \cup \mathcal{D}^{\oplus}$. To assess the *speed* of an update, we report the speed-up factor compared to retraining by dividing the time required for retraining by the time required for updating (equation 6 and 7). Retraining and updating times were recorded on an NVIDIA RTX 4090 GPU and an AMD Ryzen 9 7950X CPU, respectively.

We choose common pretrained DNN **architectures** from the literature [15,14]. For image datasets, we employ a Vision Transformer (ViT) [11] with pretrained weights via self-supervised learning, complemented by a randomly initialized fully connected layer. Specifically, we use the DINOv2-ViT-S/14 model [30] with a feature dimension of $D = 384$ in its final hidden layer. For text datasets, we employ the transformer-based pretrained language model BERT [10]. We utilize BERT-BASED-UNCASED from the Huggingface library [41] with a feature dimension of $D = 768$ and a maximum sequence length of 512. We train each DNN by finetuning for 200 epochs, employing the Rectified Adam optimizer [26] with a training batch size of 64, a learning rate of 0.01 for images and 0.1 for text, and weight decay of 0.0001. In addition, we utilize a cosine annealing learning rate scheduler. These hyperparameters were determined empirically to be effective across all datasets by investigating the loss convergence on validation splits.

4.2 Experiments

Hyperparameter Ablation: In equation 6, we introduced the hyperparameter γ, which controls the step size of our update. Intuitively, this factor determines the extent to which the DNN is influenced by the new dataset $\mathcal{D}^{\oplus}$. This factor is essential to control the update process and avoid issues such as catastrophic forgetting. Similarly, first-order and MC-based updates also utilize this factor to mitigate such problems. For further details, we refer to Appendix A.

To investigate the influence of γ and determine a suitable value for all subsequent experiments, we conduct a simple ablation study on two datasets. The results of our update are shown here, while the results for first-order and MC-based updates can be found in Appendix B. We determine the value of γ in this manner since an extensive hyperparameter search for update methods is typically impractical in an online setting [9]. Hence, fixing a value beforehand is necessary. We randomly sample an initial dataset $\mathcal{D}$ of 50 instances and train our baseline DNN. Subsequently, updates and retraining are performed on randomly sampled datasets $|\mathcal{D}^{\oplus}| \in \{1, \dots, 10\}$, and the accuracy is computed on a validation split. We repeat this process for different values of γ.

The resulting curves in Figure 2 indicate that our update with $\mathcal{D}^{\oplus}$ consistently achieves better performance than the baseline DNN that is only trained on $\mathcal{D}$. For both CIFAR-10 and DBPedia, updating with $\gamma = 1$ does not yield accuracies close to retraining, suggesting that the update is too weak. By increasing γ, we observe accuracies much closer to complete retraining, with $\gamma = 10$ being sufficient for CIFAR-10 and DBPedia. For CIFAR-10, we also notice that a very high value, i.e., $\gamma = 30$, can lead to worse performance, likely due to catastrophic forgetting. To ensure effective updates across all datasets, we will be using $\gamma = 10$ in all subsequent experiments. While this may not be optimal for some datasets, it should ensure a consistently working update in all cases.

Different Learning Stages: To investigate how our update behaves at different stages of learning, we train the baseline DNN on varying sizes of initial datasets $\mathcal{D}$ and update it with a new dataset of fixed size $|\mathcal{D}^{\oplus}| = 10$. To better visualize the differences, we report accuracy improvement of updated/retrained DNNs relative to the baseline in Figure 3. The results demonstrate that our updates provide the highest accuracy improvements across all datasets, highlighting the effective and consistent performance improvements of our update at different learning stages. While first-order and MC-based updates are also effective in earlier stages (when $|\mathcal{D}| < 50$), they tend to be less effective and even deteriorate accuracy in later stages. Compared to the first-order update, our update consistently enhances performance due to including the Hessian. As the Hessian considers curvature information about the posterior, the update is more robust regarding the choice of γ.

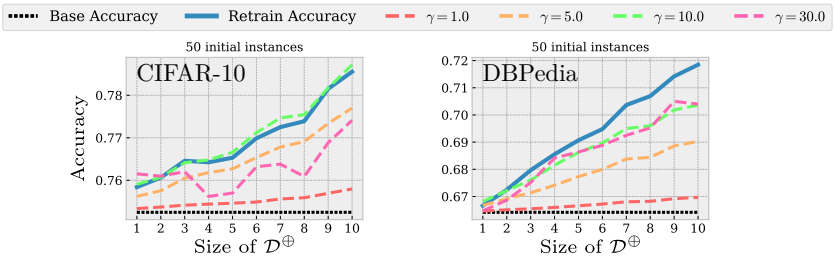

Fig. 2. Accuracies after updating with different values for γ in comparison to the baseline DNN and retraining.

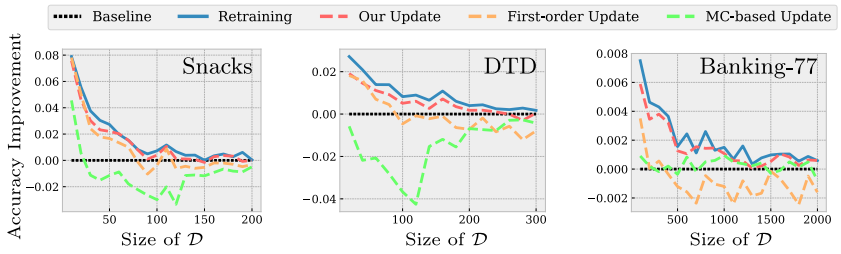

Fig. 3. Accuracy improvement curves for benchmark datasets, showing the difference in accuracy between retrained and updated DNNs for varying sizes of $\mathcal{D}$.

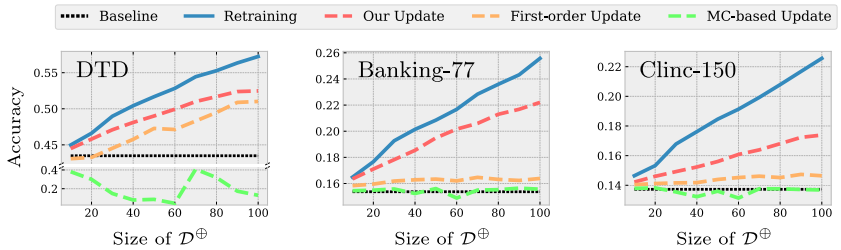

Fig. 4. Accuracy curves for three benchmark datasets after updating and retraining DNNs for varying sizes of $\mathcal{D}^{\oplus}$.

Varying Size of $\mathcal{D}^{\oplus}$: To investigate our update's behavior with an increasing number of new data points in $\mathcal{D}^{\oplus}$, we train a baseline DNN with a fixed initial dataset $|\mathcal{D}| = 100$ and vary the size of the new dataset $|\mathcal{D}^{\oplus}| \in \{10, 20, \ldots, 100\}$. We report the results for the most complex datasets DTD, Banking-77, and Clinc-150. In Figure 4, we observe that as the size of $\mathcal{D}^{\oplus}$ increases, the accuracy of retraining, our update, and the first-order update consistently improves. In contrast, MC-based updates result in worse accuracies than the baseline, indicating that it is not suited for an increasing size of $\mathcal{D}^{\oplus}$. Considering our update, we see that it consistently achieves better accuracies compared to competitors, regardless of the complexity of the dataset. Moreover, first-order updates seem to

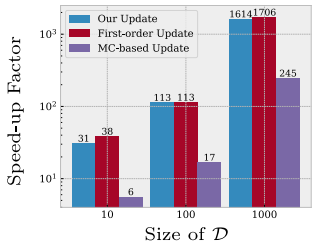

Fig. 5. Speed-up of update methods compared to retraining.

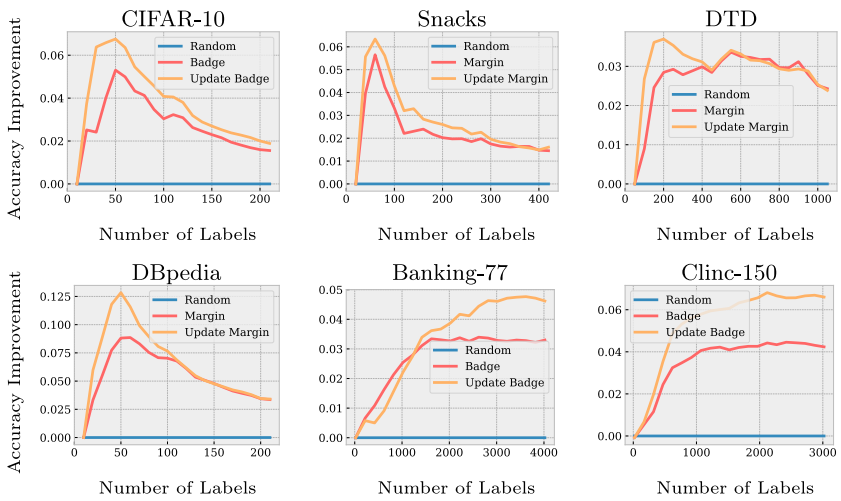

Fig. 6. Accuracy improvement curves for different datasets showing the accuracy difference between the respective selection strategy and random instance selection.

be less effective on the more complex datasets such as Banking-77 and Clinc-150, highlighting the importance of the Hessian.

Time Comparison: Finally, to evaluate the speed of updates, we fix the size of the new dataset to $|\mathcal{D}^{\oplus}| = 10$ and compute the speed-up relative to retraining by varying the initial dataset size $|\mathcal{D}|$. Figure 5 presents the speed-up factors on CIFAR-10. All update methods are faster than retraining, with the first-order update being the fastest. For example, with an initial dataset of $|\mathcal{D}| = 1000$, the first-order update is about 1700 times faster than retraining. Notably, our update provides a similar speed-up factor while yielding more effective updates by using the closed-form Hessian update. Compared to MC-based updates, both the first-order and our update are significantly faster.

5 Deep Active Learning

In this section, we examine the proposed update in AL. First, we introduce a new framework that uses our updates to exploit label information during batch construction. Essentially, this approach mimics single instance AL, in which the model is retrained after each label acquisition. Next, we employ our update to approximate an optimal look-ahead strategy. Instead of obtaining future performance of the DNN with expensive retraining, we realize this through our update. Here, we average metrics over 30 repetitions to account for reproducibility challenges in AL [29]. Labeling budgets and acquisition sizes differ based on the complexity of a dataset. A more detailed experimental setup and all learning curves, including ones that report absolute values, are available in Appendix F.

Improved Batch Selection via Updates: A naive and suboptimal way of using sequential selection strategies for batch selection is to use the top-b scoring

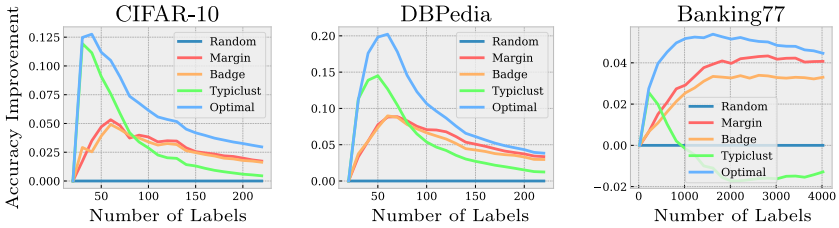

Fig. 7. Accuracy improvement over random selection of popular selection strategies compared to our upper baseline approximating optimal batch selection.

instances [18]. Our idea is to overcome the necessity of batch strategies by using the proposed update with sequential strategies as a fast alternative to retraining. Thus, we iteratively select the highest-scoring instance b times and update the DNN between each selection. After acquiring b labels, we retrain the DNN similar to batch selection strategies. An algorithm can be found in Appendix F.

The hypothesis is that already well-performing sequential selection strategies [37] can simply be used in a batch setting and that our framework can achieve higher performance compared to selecting the top-b instances. Here, we consider the widely used strategy Margin, which has proven to be effective in several studies [3]. Additionally, we are interested in whether this idea can also replace the diversity component of a batch selection strategy. Therefore, we also evaluate the popular strategy Badge [1] in combination with our updates.

Figure 6 shows the accuracy improvement curves relative to a random instance selection. The query strategies using our updates outperform the respective top-b selection strategies. Specifically, we see improved performance in early stages when redundancy within a batch plays an important role. Moreover, combining our update with Badge also results in improved accuracy. This indicates that *selecting a single instance and updating the DNN* leads to a more effective selection than using the k-MEANS++ algorithm as proposed in Badge.

Updating in Look-Ahead Strategies: The idea of look-ahead strategies is to select instances that, once labeled and added to the labeled pool, maximize the performance of the model [34]. Unlike uncertainty- or diversity-based approaches, look-ahead strategies select instances based on an optimal criterion: the model's actual performance. However, they are often neglected in deep AL due to the high computational requirements. One of the biggest bottlenecks in the selection is retraining. DNNs are not well-suited for this due to their long training process. For this reason, we employ our proposed update to make this feasible.

Here, we consider a near-optimal strategy with access to ground truth information, including labels and validation datasets. It can be considered as an upper baseline in deep AL research. For the selection, we randomly sample 2000 subsets, each with a size equal to the acquisition size, and assess how their addition to the labeled pool affects the performance. The batch leading to the highest performance gain is selected. While this approach would traditionally require 2000 times of retraining our update enables the efficient use of this strategy.

We also include the recently proposed Typiclust strategy, which demonstrated strong performance [15], especially in early stages of AL. Figure 7 presents the resulting accuracy improvement compared to random instance selection. Our optimal strategy, using updates rather than full retraining, performs exceptionally well, consistently outperforming all competitors. Based on these results, we can see that the current selection strategies still have much potential for improvement. Interestingly, we can see that Typiclust's selection in the early stages of AL seems to be close to an optimal selection but declines in effectiveness in later stages.

6 Conclusion

We proposed an efficient second-order update for DNNs in AL using the Gaussian posterior of a last-layer LA. It achieves low computational complexity through a closed-form computation of the required inverse Hessian. An extensive experimental evaluation showed that the proposed update provides an efficient alternative to retraining. Based on this, we introduced a new batch selection framework by sequentially updating the DNN after each label, offering a new perspective on constructing batches without resorting to heuristics such as clustering. Additionally, we realized a look-ahead strategy as a feasible upper baseline approximating optimal batch selection, highlighting the great potential for improvement in current research on batch selection strategies. In future work, we plan to utilize the proposed updates to enhance look-ahead selection strategies [34] in deep AL. As these strategies are based on decision-theoretic principles, they naturally balance explorative and exploitative instance selection, a key challenge in AL.

References

1. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In: ICLR (2020)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: ISWC. pp. 722–735 (2007)
3. Bahri, D., Jiang, H., Schuster, T., Rostamizadeh, A.: Is margin all you need? an extensive empirical study of active learning on tabular data. arXiv preprint arXiv:2210.03822 (2022)
4. Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., Vulić, I.: Efficient intent detection with dual sentence encoders. In: NLP4ConvAI. pp. 38–45 (2020)
5. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , Vedaldi, A.: Describing textures in the wild. In: CVPR. pp. 3606–3613 (2014)
6. Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., Zaharia, M.: Selection via proxy: Efficient data selection for deep learning. In: ICLR (2020)
7. Das, A.M., Bhatt, G., Bhalerao, M.M., Gao, V.R., Yang, R., Bilmes, J.: Accelerating batch active learning using continual learning techniques. TMLR (2023)
8. Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., Hennig, P.: Laplace redux-effortless Bayesian deep learning. In: NeurIPS (2021)

9. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. TPAMI **44**(7), 3366–3385 (2021)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
11. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
12. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: ICML. pp. 1050–1059 (2016)
13. Gal, Y., Islam, R., Ghahramani, Z.: Deep Bayesian active learning with image data. In: ICML. pp. 1183–1192 (2017)
14. Gupte, S.R., Aklilu, J., Nirschl, J.J., Yeung-Levy, S.: Revisiting Active Learning in the Era of Vision Foundation Models. TMLR (2024)
15. Hacohen, G., Dekel, A., Weinshall, D.: Active learning on a budget: Opposite strategies suit high and low budgets. In: ICML. pp. 8175–8195 (2022)
16. Hoi, S.C.H., Sahoo, D., Lu, J., Zhao, P.: Online learning: A comprehensive survey. Neurocomputing **459**, 249–289 (2021)
17. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. PNAS **114**(13), 3521–3526 (2017)
18. Kirsch, A., Farquhar, S., Atighehchian, P., Jesson, A., Branchaud-Charron, F., Gal, Y.: Stochastic Batch Acquisition: A Simple Baseline for Deep Active Learning. TMLR (2023)
19. Kirsch, A., Kossen, J., Gal, Y.: Marginal and Joint Cross-Entropies & Predictives for Online Bayesian Inference, Active Learning, and Active Sampling. arXiv preprint arXiv:2205.08766 (2022)
20. Kirsch, A., Van Amersfoort, J., Gal, Y.: BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. In: NeurIPS (2019)
21. Kivinen, J., Smola, A., Williamson, R.C.: Online learning with kernels. In: NeurIPS (2001)
22. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Toronto (2009)
23. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: NeurIPS (2017)
24. Larson, S., Mahendran, A., Peper, J.J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J.K., Leach, K., Laurenzano, M.A., Tang, L., Mars, J.: An evaluation dataset for intent classification and out-of-scope prediction. In: EMNLP. pp. 1311–1316 (2019)
25. Liu, J.Z., Padhy, S., Ren, J., Lin, Z., Wen, Y., Jerfel, G., Nado, Z., Snoek, J., Tran, D., Lakshminarayanan, B.: A simple approach to improve single-model deep uncertainty via distance-awareness. JMLR **24**(42), 1–63 (2023)
26. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond. In: ICLR (2019)
27. Lu, Z., Ie, E., Sha, F.: Mean-field approximation to Gaussian-softmax integral with application to uncertainty estimation. arXiv preprint arXiv:2006.07584 (2020)
28. Matthijs: Snacks dataset. `https://huggingface.co/datasets/Matthijs/snacks` (2021), accessed: 2024-05-20
29. Munjal, P., Hayat, N., Hayat, M., Sourati, J., Khan, S.: Towards robust and reproducible active learning using neural networks. In: CVPR. pp. 223–232 (2022)

30. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: DINOv2: Learning Robust Visual Features without Supervision. TMLR (2023)
31. Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., Snoek, J.: Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In: NeurIPS (2019)
32. Rauch, L., Assenmacher, M., Huseljic, D., Wirth, M., Bischl, B., Sick, B.: ActiveGLAE: A Benchmark for Deep Active Learning with Transformers. In: ECML (2023)
33. Ritter, H., Botev, A., Barber, D.: Online structured Laplace approximations for overcoming catastrophic forgetting. In: NeurIPS (2018)
34. Roy, N., McCallum, A.: Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In: ICML. pp. 441–448 (2001)
35. Sahoo, D., Pham, Q., Lu, J., Hoi, S.C.: Online deep learning: learning deep neural networks on the fly. In: IJCAI. pp. 2660–2666 (2018)
36. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. In: ICLR (2018)
37. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
38. Spiegelhalter, D.J., Lauritzen, S.L.: Sequential updating of conditional probabilities on directed graphical structures. Networks **20**(5), 579–605 (1990)
39. Tan, W., Du, L., Buntine, W.: Diversity enhanced active learning with strictly proper scoring rules. In: NeurIPS (2021)
40. Wang, H., Yeung, D.Y.: A survey on Bayesian deep learning. CSUR **53**(5), 1–37 (2020)
41. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.: Transformers: State-of-the-art natural language processing. In: EMNLP. pp. 38–45 (2020)
42. Woodbury, M.A.: Inverting modified matrices. Department of Statistics, Princeton University (1950)
43. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. In: ICLR (2018)
44. Zhang, J., Chen, Y., Canal, G., Das, A.M., Bhatt, G., Mussmann, S., Zhu, Y., Bilmes, J., Du, S.S., Jamieson, K., Nowak, R.D.: Labelbench: A comprehensive framework for benchmarking adaptive label-efficient learning. DMLR (2024)
45. Zhao, G., Dougherty, E., Yoon, B.J., Alexander, F., Qian, X.: Uncertainty-aware active learning for optimal Bayesian classifier. In: ICLR (2021)
46. Zinkevich, M.: Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In: ICML. pp. 928–936 (2003)