Active Learning Guided by Efficient Surrogate Learners

Yunpyo An[*1], **Suyeong Park**[*1], **Kwang In Kim**[2]

[1]UNIST
[2]POSTECH
{anyunpyo,suyeong}@unist.ac.kr, kimkin@postech.ac.kr

Abstract

Re-training a deep learning model each time a single data point receives a new label is impractical due to the inherent complexity of the training process. Consequently, existing active learning (AL) algorithms tend to adopt a batch-based approach where, during each AL iteration, a set of data points is collectively chosen for annotation. However, this strategy frequently leads to redundant sampling, ultimately eroding the efficacy of the labeling procedure. In this paper, we introduce a new AL algorithm that harnesses the power of a Gaussian process surrogate in conjunction with the neural network principal learner. Our proposed model adeptly updates the surrogate learner for every new data instance, enabling it to emulate and capitalize on the continuous learning dynamics of the neural network without necessitating a complete retraining of the principal model for each individual label. Experiments on four benchmark datasets demonstrate that this approach yields significant enhancements, either rivaling or aligning with the performance of state-of-the-art techniques.

1 Introduction

The success of deep learning heavily relies on a substantial amount of labeled data. However, creating extensive datasets poses challenges, particularly for problems demanding significant labeling efforts. As a result, the utilization of deep learning is constrained to domains with feasible labeling resources. Active learning (AL) seeks to surpass this constraint by strategically selecting the most informative data instances for labeling within a predetermined labeling budget (Ren et al. 2020; Settles 2009; Aggarwal et al. 2015).

Typically, in AL, a baseline learner is initially provided with a dataset where only a small or no subset is labeled. Throughout the learning process, AL algorithms analyze the data distribution and the learner's progress to recommend specific data instances for labeling. The effectiveness of AL hinges on accurately identifying both difficult (or *uncertain*) data points, as well as those wielding substantial *influence* over the learner's overall decisions once labeled. Identifying influential data requires capturing the global shape of the underlying data distribution. For this, existing AL algorithms primarily focus on achieving *diversity* by populating areas where labels are sparsely sampled. This approach is particularly effective in the early stages of AL when the number of labeled instances is small and the learner's predictions are unreliable. However, its main limitation is the lack of consideration for the learner's progress. As more labels are acquired, the learner becomes a more reliable estimator of the underlying ground truth. In such cases, prioritize regions where the learner struggles could be more advantageous than covering areas of already established confidence.

To address this limitation, it is also essential to identify instances where the learner's predictions require deeper investigation. This entails selecting instances with the high degree of uncertainty in the learner's predictions. However, in the early learning stages, the learner may not have a comprehensive understanding of the problem, and uncertainty estimates can be unreliable. Another significant challenge is the resulting redundancy in labeling. Given the computational intensity of training neural networks, updating the model for each new data instance is impracticable. Consequently, prevailing AL algorithms often adopt a batch-based strategy, wherein each AL iteration involves collectively choosing a set of data points for annotation. If not carefully managed, this can result in spatial aggregations where uncertainty spreads to neighboring instances. However, addressing uncertainties within these aggregations can frequently be achieved by annotating only one or a handful of instances and subsequently retraining the learner. Existing approaches have attempted to address this issue through additional data diversification techniques (Nguyen and Smeulder 2004; Sinha et al. 2019).

In this paper, we introduce a novel AL algorithm that integrates these two essential objectives into a unified approach, using a single surrogate model for the neural network learner. Our algorithm operates in a batch mode, wherein the primary neural network learner is trained exclusively when a batch of labels is amassed. However, it mimics the continuous learning trajectory akin to that of the neural learner by harnessing an efficient Gaussian process (GP) learner, which is refreshed each time a new label is incorporated. By adopting well-established Bayesian techniques, our algorithm provides a rigorous framework for effectively and efficiently identifying influential points: At each stage, a new data instance is selected to maximize the resulting information gain over the entire dataset. Furthermore, our model rapidly identifies the most uncertain data points by

[*]These authors contributed equally.

instantaneously updating the surrogate GP learner. When a new data point is added, the confidence of predictions for its spatial neighbors is immediately improved, eliminating the need for additional data diversification.

In experiments on four datasets, our algorithm consistently outperforms or performs comparably to state-of-the-art approaches in both the early and later learning stages.

2 Related Work

Existing work on active learning has focused on identifying diverse points (Sener and Savarese 2018; Aodha et al. 2014; Nguyen and Smeulder 2004), uncertain points (Yoo and Kweon 2019; Tong and Koller 2001; Yun et al. 2020; Kirsch et al. 2019), and their combinations (Ash et al. 2020; Elhamifar et al. 2013). Diversity-based approaches aim to efficiently cover the underlying data distribution. This can be achieved through e.g. pre-clustering of unlabeled data (Nguyen and Smeulder 2004), predicting the impact of labeling individual instances on predictions (Aodha et al. 2014), or maximizing mutual information between labeled and unlabeled instances (Guo 2010). Sener and Savarese (2018) introduced a core-set approach that minimizes the upper bound on generalization error, resulting in minimum coverings of data space through labeled data. Geifman and El-Yaniv (2017) proposed a sequential algorithm that selects the farthest traversals from labeled instances to promote diversity. Gissin and Shalev-Shwartz (2019) also enforced diversity by selecting data points that make labeled and unlabeled points indistinguishable. Sinha et al. (2019)'s variational adversarial active learning (VAAL) constructs a latent space using a variational autoencoder and an adversarial network to achieve sample diversity. While diversity-based approaches effectively select representative data points, they may not fully exploit the information acquired by the baseline learner in the task: At each round, assessing the trained learner on unlabeled data can offer insights into areas where additional labeling is desired, e.g. areas close to the decision boundaries.

Uncertainty-based approaches focus on identifying areas where the learner's predictions are uncertain. Tong and Koller (2001)'s method selects instances close to the decision boundary of a support vector machine learner. Maximizing the entropy of the predictive class-conditional distribution is a common approach for uncertainty-based selection (Yun et al. 2020). For Bayesian learners, Houlsby et al. (2011) introduced the Bayesian active learning by disagreement (BALD) algorithm, which selects instances based on information gain. Kirsch et al. (2019) extended BALD to deep learning with BatchBALD, suggesting multiple instances for labeling simultaneously. Tran et al. (2019) further extended BALD by incorporating generative models to synthesize informative instances. Yoo and Kweon (2019)'s learning loss algorithm trains a separate module to predict learner losses and selects instances with the highest predicted losses. Uncertainty-based approaches are particularly effective when the learner's predictions closely approximate the underlying ground truth. However, their performance can suffer at early active learning stages when the learner's predictions are unreliable.

Elhamifar et al. (2013) proposed a hybrid approach that integrates label diversity and the learner's predictive uncertainty into a convex optimization problem. Zhang et al. (2020)'s state-relabeling adversarial active learning (SRAAL) extends VAAL to incorporate model uncertainty. This algorithm is specifically designed for learners that share their latent space with variational autoencoders. Ash et al. (2020) developed the batch active learning by diverse gradient embeddings (BADGE) method, which balances diversity and uncertainty by analyzing the magnitude of loss gradients for candidate points and their distances to previously labeled points. Kim et al. (2021) presented the task-aware VAAL (TA-VAAL) algorithm, which extends VAAL by incorporating predicted learner losses. Caramalau et al. (2021) proposed a sequential graph convolutional network (GCN)-based algorithm that improves uncertainty sampling by analyzing the overall distribution of data using graph embeddings of data instances. Ash et al. (2021) optimized the bound on maximum likelihood estimation error using Fisher information for model parameters. In our experiments, we demonstrate that our method outperforms or achieves comparable performance to the state-of-the-art BADGE, TA-VAAL, and sequential GCN algorithms.

Our algorithm shares a connection with Coleman et al. (2020)'s proxy-based approach, which leverages a compact learner network to enhance the speed of the label selection process. Notably, our approach distinguishes itself through the incorporation of Bayesian GP surrogates. Although Coleman et al. (2020)'s algorithm marks a significant stride forward in enhancing selection speed, it remains deficient in fulfilling the computational efficiency criteria essential for incremental labeling. For CIFAR10, this algorithm necessitates 83 hours to label a mere 1,000 data points. Also, unlike (Coleman et al. 2020), which exclusively relies on model uncertainty, our algorithm harnesses the power of a Bayesian learner to directly combine both the global influence of labeling and the inherent model uncertainty.

3 Active Learning Guided by Gaussian Process Surrogate Learners

In traditional supervised learning, a function $f : \mathcal{X} \to \mathcal{Y}$ is learned based on a labeled training set $T = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\} \subset \mathcal{X} \times \mathcal{Y}$, sampled from the joint distribution of $\mathcal{X}$ and $\mathcal{Y}$. In active learning (AL), however, we initially have access only to an input dataset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. The AL algorithm is then given a budget to suggest B data instances to be labeled. The output of AL is a labeled index subset L of size B that specifies the selected elements in T. In this work, we focus on classification problems, assuming that the data are one-hot encoded, with $\mathcal{Y} \subset \mathbb{R}^C$, where C is the number of classes. The baseline learner f produces probabilistic outputs, such that $\|f(\mathbf{x})\|_1 = 1$ and $[f(\mathbf{x})]_j \geq 0$ with $[f(\mathbf{x})]_j$ being the j-th element (corresponding to the j-th class) of $f(\mathbf{x})$. We will use f to denote both the baseline learning algorithm and its output, forming a classification function.

Our approach follows an incremental strategy for constructing L. Starting from an initial set of labeled points L^0,

at each stage t, L^t is expanded by adding a single label index l^t: $L^{t+1} = L^t \cup \{l^t\}$. This process is guided by a utility function $u : \{1, \ldots, N\} \to \mathbb{R}$ such that l^t is chosen as the maximizer of u among the indices in $\{1, \ldots, N\} \setminus L^t$.

A good utility function should encompass two aspects: 1) the difficulty (or *uncertainty*) associated with classifying each data instance, and 2) the *influence* that labeling a data instance has on improving the classification decisions for other instances. However, realizing these objectives necessitates the ability to continuously observe how the baseline learner f evolves at each stage (denoted as f^t, trained on L^t). Selecting B data instances simultaneously at a single stage, based solely on their utility values, often leads to redundant aggregations of spatial neighbors. For instance, if a data instance $\mathbf{x}_i$ has the highest utility value $u(i)$, it is likely that its spatial neighbors also exhibit similarly high utility values. Labeling the entire spatial aggregation can be redundant. Applying this continuous retraining strategy becomes challenging when f is a deep neural network (DNN) due to prohibitively high computational costs. Existing approaches circumvent this challenge by either designing utilities that are independent of the learner f (Sener and Savarese 2018; Aodha et al. 2014; Nguyen and Smeulder 2004), or by employing auxiliary processes to promote spatial diversity in labeled data (Kirsch et al. 2019; Nguyen and Smeulder 2004; Sinha et al. 2019). The latter often requires fine-tuning hyperparameters and heuristics to balance the selection of difficult labels with retaining diversity.

Our algorithm trains a computationally efficient surrogate learner $\hat{f}$ in parallel, which simulates the continuous learning behavior of the baseline f. We use a Gaussian process (GP) estimator for this purpose, allowing us to leverage well-established Bayesian inference techniques in designing and efficiently evaluating the utility function u. Notably, our approach does not necessitate additional mechanisms to promote label diversity. When a data instance with high utility is labeled, the surrogate $\hat{f}$ is instantly updated, leading to the corresponding suppression of utilities for its neighbors.

Gaussian Process Surrogate Learner: For a given budget B, the DNN learner f is trained only at every I-th stage. To capture the behavior of f between these stages, we employ a continuously updated surrogate GP learner $\hat{f}$. At each I-th stage, our surrogate $\hat{f}$ is initialized to match f (with softmax applied to the output layer). Between these I-th stages, $\hat{f}$ undergoes training using the accumulated labels. Specifically, for each newly selected data instance $\mathbf{x}$ to be labeled, the training label for the GP is computed as $\mathbf{y}$ and subsequently, the prediction from $f(\mathbf{x})$ is subtracted. We will use two types of utility functions to represent the uncertainty and influence of the predictions made by f and $\hat{f}$.

Suppose that t data instances have already been labeled at stage t. Without loss of generality, we consider these labeled points to correspond to the first t points in X: $T^t = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^t$ forms the labeled training set, while $U^t = \{\mathbf{x}_i\}_{i=t+1}^N$ represents the unlabeled set, i.e. $L^t = \{1, \ldots, t\}$.

Kernels: Our GP prior is constructed by combining Gaussian kernels based on the inputs $\mathbf{x}$ and $\mathbf{x}'$, as well as the corresponding outputs of the latest learner $f(\mathbf{x})$ and $f(\mathbf{x}')$:

$$k(\mathbf{x}, \mathbf{x}', f(\mathbf{x}), f(\mathbf{x}')) = k_{\mathbf{x}}(\mathbf{x}, \mathbf{x}')k_{\mathbf{y}}(f(\mathbf{x}), f(\mathbf{x}')), \quad (1)$$
$$k_{\mathbf{x}}(\mathbf{x}, \mathbf{x}') = \overline{k}(\mathbf{x}, \mathbf{x}', \sigma_{\mathbf{x}}^2), k_{\mathbf{y}}(\mathbf{y}, \mathbf{y}') = \overline{k}(\mathbf{y}, \mathbf{y}', \sigma_f^2),$$
$$\overline{k}(\mathbf{a}, \mathbf{a}', b) = \exp\left(-\frac{\|\mathbf{a} - \mathbf{a}'\|^2}{b}\right),$$

where $\sigma_{\mathbf{x}}^2$ and σ_f^2 are hyperparameters controlling the kernel widths. The output kernel $k_{\mathbf{y}}$ values are calculated using the stored learner values $f(\mathbf{x}_i)$ at the beginning of each training interval of size I (see Algorithm 1 and Sec. 4). The use of this product kernel allows the resulting surrogate $\hat{f}$ to accurately capture the behavior of the underlying deep learner f, while preserving the class boundaries formed by f without excessive smoothing. When training a separate baseline network with 4,500 labeled samples (for the FashionMNIST dataset; see Sec. 4 for more details), experiments conducted on 5 different random initializations showed that using the combined kernel $k(\mathbf{x}, \mathbf{x}', f(\mathbf{x}), f(\mathbf{x}'))$ reduced the mean absolute deviation between $\hat{f}$ and f on the training set by 34%, compared to using only the standard input kernel $k_{\mathbf{x}}(\mathbf{x}, \mathbf{x}')$.

Predictive Model: Our GP learner $\hat{f}$ employs an i.i.d. Gaussian likelihood across all classes and data instances (Rasmussen and Williams 2006): $[\mathbf{y}]_j \sim \mathcal{N}([f(\mathbf{x})]_j, \sigma^2)$, where $\mathcal{N}(\mu, \sigma^2)$ represents a Gaussian distribution with mean μ and variance σ^2. By combining this likelihood with the GP prior (Eq. 1), the prediction of $\hat{f}$ for an unlabeled point $\mathbf{x}_i \in U^t$ is represented as an isotropic Gaussian random vector of size C:

$$p(\mathbf{y}_i | T^t, \mathbf{x}_i) = \mathcal{N}(\boldsymbol{\mu}_i^t, \boldsymbol{\Sigma}_i^t), \text{ where} \quad (2)$$
$$\boldsymbol{\mu}_i^t = (\mathbf{k}_i^t)^\top (\mathbf{K}^t + \sigma^2 \mathbf{I})^{-1} \mathbf{Y}^t,$$
$$\boldsymbol{\Sigma}_i^t = (1 - (\mathbf{k}_i^t)^\top (\mathbf{K}^t + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_i^t) \mathbf{I}.$$

Here, $[\mathbf{K}^t]_{mn} = k(\mathbf{x}_m, \mathbf{x}_n, f(\mathbf{x}_m), f(\mathbf{x}_n))$, $[\mathbf{k}_i^t]_m = k(\mathbf{x}_i, \mathbf{x}_m, f(\mathbf{x}_i), f(\mathbf{x}_m))$ for $1 \le m, n \le t$, $\mathbf{Y}^t = [\mathbf{y}_1^\top, \ldots, \mathbf{y}_t^\top]^\top$, and $[\mathbf{K}^t]_{mn}$ is the (m, n)-th element of $\mathbf{K}^t$. This model requires inverting the kernel matrix $\mathbf{K}^t$ of size $t \times t$ which grows quickly as AL progresses. To ensure that the computational complexity of $\hat{f}$ predictions (Eq. 2) remains manageable, we employ a sparse GP approximation (Snelson and Ghahramani 2006) for our product kernel k using two sets of basis points $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_K\}$ and $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$:

$$\boldsymbol{\mu}_i^t \approx (\mathbf{k}_i^t)^\top (\mathbf{Q}^t)^{-1} \mathbf{K}_{XP}^t (\boldsymbol{\Lambda}^t + \sigma^2 \mathbf{I})^{-1} \mathbf{Y}^t, \quad (3)$$
$$\boldsymbol{\Sigma}_i^t \approx (1 - (\mathbf{k}_i^t)^\top (\mathbf{K}_{PP}^{-1} - (\mathbf{Q}^t)^{-1}) \mathbf{k}_i^t) \mathbf{I} + \sigma^2 \mathbf{I}, \text{ where}$$
$$\mathbf{Q}^t = \mathbf{K}_{PP} + (\mathbf{K}_{XP}^t)^\top (\boldsymbol{\Lambda}^t + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_{XP}^t,$$

$[\mathbf{K}_{XP}^t]_{mn} = k(\mathbf{x}_m, \mathbf{u}_n, f(\mathbf{x}_m), \mathbf{v}_n)$, $[\mathbf{K}_{PP}]_{mn} = k(\mathbf{u}_m, \mathbf{u}_n, \mathbf{v}_m, \mathbf{v}_n)$, and $[\mathbf{k}_i^t]_m = k(\mathbf{x}_i, \mathbf{u}_m, f(\mathbf{x}_i), \mathbf{v}_m)$ for $1 \le m, n \le t$, and $\boldsymbol{\Lambda}^t$ is a diagonal matrix with its n-th entry $[\boldsymbol{\Lambda}^t]_{nn}$ defined as $1 - (\mathbf{k}_n^t)^\top \mathbf{K}_{PP}^{-1} \mathbf{k}_n^t$. The basis points U are obtained by extracting the cluster centers of X using K-means clustering, while V is randomly sampled as C-dimensional probability simplices. The complexity of evaluating a prediction (Eq. 3) now scales linearly with the number of labeled data points: $\mathcal{O}(tK^2)$.

Influence-Based Utility u^1: As a Bayesian model, the output of $\hat{f}$ for an unlabeled input $\mathbf{x}_i$ is presented as a probability distribution $p(\mathbf{y}_i|T^t, \mathbf{x}_i)$, which naturally quantifies uncertainty. The diagonal elements of the predictive covariance matrix for input $\mathbf{x}_i$ correspond to the entropies of the predictions for each class.[1] Therefore, the trace of this matrix indicates the overall uncertainty of the current model's prediction for $\mathbf{x}_i$.

However, relying solely on individual data point entropies fails to capture their *influence* when labeled, and optimizing based on this criterion alone tends to prioritize outliers. This behavior arises because (the diagonal terms of) the predictive covariance matrix $\mathbf{\Sigma}_i^t$ tends to increase as the corresponding unlabeled points deviate from the labeled training set: See (Sollich and Williams 2005) for an analysis of this behavior for large-scale problems. For a more meaningful measure of utility, we define $u^1(i)$ based on the reduction in predictive entropies of the *entire remaining unlabeled set* U^{t-1} upon labeling $\mathbf{x}_i$. To achieve this, we maintain the predictive covariance of the unlabeled set at each stage, storing only a single diagonal entry per data point due to the isotropic nature of the covariance estimates in our model:

$$\begin{aligned} u^1(i) &= \sum_{j=t+1}^{N} \text{trace}[\mathbf{\Sigma}_j^t] - \sum_{j=t+1}^{N} \text{trace}[\mathbf{\Sigma}_j(i)] \\ &= C \sum_{j=t+1}^{N} (\mathbf{k}_j^t)^\top (\mathbf{Q}(i)^{-1} - (\mathbf{Q}^t)^{-1}) \mathbf{k}_j^t, \end{aligned}$$

where $\mathbf{\Sigma}_j(i)$ denotes the covariance for $\mathbf{x}_j \in U^{t-1} \setminus \{\mathbf{x}_i\}$ predicted by the model trained on T^t and $\mathbf{x}_i$ (assuming that $\mathbf{x}_i$ is labeled),

$$\begin{aligned} \mathbf{Q}(i) =& \mathbf{K}_{PP} + \\ & \begin{pmatrix} \mathbf{K}_{XP}^t \\ (\mathbf{k}_i^t)^\top \end{pmatrix}^\top \left(\begin{pmatrix} \mathbf{\Lambda}^t & 0 \\ 0 \lambda_i & \end{pmatrix} + \sigma^2 \mathbf{I} \right)^{-1} \begin{pmatrix} \mathbf{K}_{XP}^t \\ (\mathbf{k}_i^t)^\top \end{pmatrix}, \end{aligned} \tag{4}$$

and $\lambda_i = 1 - (\mathbf{k}_i^t)^\top \mathbf{K}_{PP}^{-1} \mathbf{k}_i^t$. Directly calculating $u(i)$ requires $\mathcal{O}(tK^2 + K^3)$-time which is prohibitive even for moderately-sized datasets. A computationally efficient solution is obtained by observing that the difference between $\mathbf{Q}(i)$ and $\mathbf{Q}^t$ is rank one in that

$$\mathbf{Q}(i) = \mathbf{Q}^t + \mathbf{a}\mathbf{a}^\top \text{ with } \mathbf{a} = \frac{\mathbf{k}_i^t}{\sqrt{\lambda_i + \sigma^2}}. \tag{5}$$

Applying the *Sherman–Morrison–Woodbury* matrix identity (Schott 2016) to Eq. 4 using Eq. 5, we obtain

$$u^1(i) = C \sum_{j=t+1}^{N} (\mathbf{k}_j^t)^\top \left(\frac{(\mathbf{Q}^t)^{-1} \mathbf{k}_i^t (\mathbf{k}_i^t)^\top (\mathbf{Q}^t)^{-1}}{\lambda_i + \sigma^2 + (\mathbf{k}_i^t)^\top (\mathbf{Q}^t)^{-1} \mathbf{k}_i^t} \right) \mathbf{k}_j^t.$$

With this form, the utility $u^1(i)$ can be evaluated in $\mathcal{O}(K^2)$-time when $(\mathbf{Q}^t)^{-1}$ is provided. Once the inverse $(\mathbf{Q}^0)^{-1}$ is explicitly computed at stage 0, the inverse $(\mathbf{Q}^t)^{-1}$ at each subsequent stage t can be calculated from $(\mathbf{Q}^{t-1})^{-1}$ in $\mathcal{O}(K^2)$ time.

Discussion: Our approach draws inspiration from Bayesian AL methods that aim to minimize the entropy of the learner's parameters and their approximations(e.g. (Houlsby et al. 2011; Kirsch et al. 2019)). However, unlike these approaches, we minimize the entropy over predictions made on the entire dataset, rather than focusing on model parameters. This allows us to calculate entropy independently of the specific parametric forms of the learner, and it enables us to use a GP as a surrogate for the deep learner f.

Evaluating u^1 does not require knowing the actual label of each candidate point $\mathbf{x}_i$, even though u^1 was derived under the assumption that $\mathbf{x}_i$ is labeled. This property arises from the i.i.d. Gaussian noise model. In general, for an underlying classification function $\tilde{f}$, the likelihood $p(\mathbf{y}|\tilde{f}(\mathbf{x}))$ of observing data $\mathbf{y}$ given an input $\mathbf{x}$ is not Gaussian. In such cases, logistic likelihood models are commonly employed in GP models. However, these models yield covariance predictions that explicitly depend on the labels of each candidate point, while the labels become available only after the corresponding data points are actually selected.

Our utility u^1 uses the expected reduction of predictive variances when labeled. Theoretically, a more appealing approach might be to use the expected reduction of test errors. However, since ground-truth labels are not available, such error reduction cannot be directly calculated. Existing approaches therefore introduced certain model assumptions (Freytag et al. 2014; Vijayanarasimhan and Kapoor 2010) (which might hold for only specific learners).

Uncertainty-Based Utility u^2: This is based on the entropies of the class-conditional predictive distributions generated by DNNs. This differs from the entropy used in u^1, which is defined for continuous GP predictive distributions. A straightforward approach to design such a utility is to directly measure the entropy of f-prediction at each stage:

$$\widehat{u}^2(i) = \text{Ent}(f^t(\mathbf{x}_i)), \tag{6}$$

where $\text{Ent}(\mathbf{p})$ represents the entropy of a distribution $\mathbf{p}$.

Maximizing $\widehat{u}^2(i)$ effectively selects the most uncertain point for classification. However, implementing this strategy requires retraining f^t at each stage, which is infeasible due to the high training cost. Instead, we train f only at every I-th stage and use the Gaussian process (GP) surrogate to estimate entropy values for the intermediate stages. Let's consider stage s, where the DNN classifier f^s is newly trained, and we calculate the corresponding entropy values $\{\text{Ent}(f^s(\mathbf{x}_{t+1})), \ldots, \text{Ent}(f^s(\mathbf{x}_N))\}$. The entropy values at intermediate stages are generated by calibrating the original DNN entropy values $\{\text{Ent}(f^s(\mathbf{x}_i))\}$ using the surrogate predictions. Initially, we set the calibrated entropy $\widetilde{\text{Ent}}^s(f(\mathbf{x}_i))$ at stage s as $\text{Ent}(f^s(\mathbf{x}_i))$. The new entropy at stage $t > s$ is then calculated as follows:

$$u^2(i) = \widetilde{\text{Ent}}^t(f(\mathbf{x}_i)) = \widetilde{\text{Ent}}^{t-1}(f(\mathbf{x}_i)) \frac{\text{Ent}(\rho(\hat{f}^t(\mathbf{x}_i)))}{\text{Ent}(\rho(\hat{f}^{t-1}(\mathbf{x}_i)))},$$

where $\rho(\hat{f}^t(\mathbf{x}_i))$ is the softmax output entropy of the mean prediction $\boldsymbol{\mu}_i^t$ made by the GP surrogate $\hat{f}^t$. Annotating a

[1] The variance of a Gaussian distribution is proportional to its entropy.

Algorithm 1: Active learning guided by GP proxies.

Input: Input data X, budget B, training interval I, and initial label set L^0

Output: Label index set L^B

for $t = 0, \ldots, B$ **do**
  Calculate GP predictions and update $\mathbf{Q}^t$ (Eq. 3);
  if $mod(t, I) = 0$ **then**
    Train f^t, and calculate $\{\text{Ent}(\rho(f^t(\mathbf{x}_i)))\}$ and $\{k_{\mathbf{y}}(f^t(\mathbf{x}_i), \mathbf{v}_j)\}$ (Eq. 1);
  end if
  Evaluate u^1 and u^2;
  Calculate the test accuracy estimate $P(f^t)$;
  Generate u by combining u^1 and u^2 using $P(f^t)$ (Eq. 7);
  $l^t = \arg\max u$;
  $L^t = L^{t-1} \cup \{l^t\}$;
end for

data instance $\mathbf{x}_i$ not only reduces the uncertainty of GP predictions at $\mathbf{x}_i$ but also influences its neighboring points. As a result, the calibrated entropies reflect the continuous reduction of uncertainties caused by the introduction of new labeled points during the intermediate stages.

Discussion. We also investigated the possibility of introducing a utility u^3 as an alternative to u^2, based on the reduction of predictive entropies for all data instances in the unlabeled set when a candidate point $\mathbf{x}_i$ is labeled. However, since this measure involves the label of each candidate $\mathbf{x}_i$, direct evaluation is not possible. Instead, for each class $j \in \{1, \ldots, C\}$, we tentatively assigned the corresponding class label to $\mathbf{x}_i$ and computed the resulting entropy reduction. The final utility $u^3(i)$ was obtained by averaging these hypothetical entropy values, weighted by the corresponding class probabilities $[f(\mathbf{x}_i)]_j$ predicted by the learner. In our preliminary experiments on the FashionMNIST dataset (see Sec. 4), we observed that the original utility u^2 achieved a final classification accuracy that was, on average, only 0.16% lower than that of u^3, while being approximately 50 times faster. The main computational bottleneck in evaluating u^3 was the computation of entropy values for the entire unlabeled set for each hypothesized candidate. The competitive performance of u^2 can be attributed to the fact that maximizing u^2 does not select outliers, unlike the use of predictive entropies in u^1. When employing monotonically increasing activations such as sigmoid and ReLU, DNN predictions tend to be overly confident on outliers, which are points that deviate significantly from the training set. While this artifact is generally not desirable for classification purposes, it has a favorable side-effect in AL, as outliers are assigned low class-conditional entropy values and are not selected.

Combining u^1 and u^2: Our utility functions, u^1 and u^2, exhibit complementary strengths. u^1 quantifies the overall reduction of uncertainty across the entire unlabeled set, providing an effective approach for exploring the data space, especially when the learner lacks sufficient information about the problem. It is particularly useful in scenarios where the learner's accuracy is low due to limited labeled data. However, u^1 is agnostic to the specific task at hand, as the predictive covariance $\mathbf{\Sigma}_i^t$ is solely determined by the distribution of input data instances in X and remains independent of the acquired labels (Eq. 3). On the other hand, u^2 capitalizes on the label information captured by the entropy of $f(\mathbf{x})$. However, in cases where the performance of the learner f is limited, the estimated entropies themselves can be unreliable indicators. To combine the strengths of u^1 and u^2, we employ a convex combination:

$$u(i) = (1 - P(f^t))\overline{u}^1(i) + P(f^t)\overline{u}^2(i), \tag{7}$$

where $\overline{u}^1$ and $\overline{u}^2$ represent the normalized versions of u^1 and u^2, respectively, based on their respective standard deviations computed on the entire dataset. Here, $P(f^t) \in [0, 1]$ is an estimate of the test accuracy. As we do not have access to test data, we instead treat the newly labeled training data point at stage t as a single test point before it is added to L^{t-1} and accumulate the resulting accuracies from the initial AL stage $t = 0$. Algorithm 1 summarizes the proposed algorithm.

Approximation Quality of the GP Proxies: By design, our GP surrogate $\hat{f}$ coincides with f at every I-th stage, while it may exhibit deviations from f in between these stages. However, empirical observations indicate that $\hat{f}$ effectively captures the behavior of f thanks to the product kernel k (see Eq. 1). To validate this, we trained a separate neural network learner f' at an intermediate stage with 2,500 labels (for the FashionMNIST dataset). The signal-to-noise ratio of $\hat{f}$ compared to f' at $t = 2,500$ was measured to be 15.49dB. This noise level is comparable to the variations observed in f' resulting from retraining the DNN learners using the same training data but with random initializations.

Hyperparameters and Time Complexity: Our algorithm involves three hyperparameters. The number of basis points K for U and V (Eq. 3) is fixed at 500, balancing between computational complexity and approximation accuracy of GP predictions. The input kernel parameter $\sigma_{\mathbf{x}}$ is set to 0.5 times the average distance between data instances in X, while the output kernel parameter σ_f is determined as the number of classes per dataset. The noise level σ^2 (Eq. 2) is kept small at 10^{-10}. These hyperparameters were chosen without using problem- or dataset-specific information. While fine-tuning them for each task and dataset could potentially improve performance, it would require additional validation labels, which are often scarce in AL scenarios.

The computational complexity of each iteration of our algorithm is $\mathcal{O}(K^2 \times N)$, where N is the number of unlabeled data instances and K is the rank of the sparse GP approximation (Eq. 3). On average, our algorithm takes approximately 0.35 seconds to suggest a point for labeling on the FashionMNIST dataset. In comparison, the run-times of other methods such as *VAAL*, *CoreSet*, *LearningLoss*, *BADGE*, *WS*, *SeqGCN*, and *TA-VAAL* (see Sec.4) were 23.73, 0.05, 0.12, 0.58, 0.03, 0.86, and 23.78 seconds, respectively. While instantiating continuous learning, our approach incurs comparable computational costs.

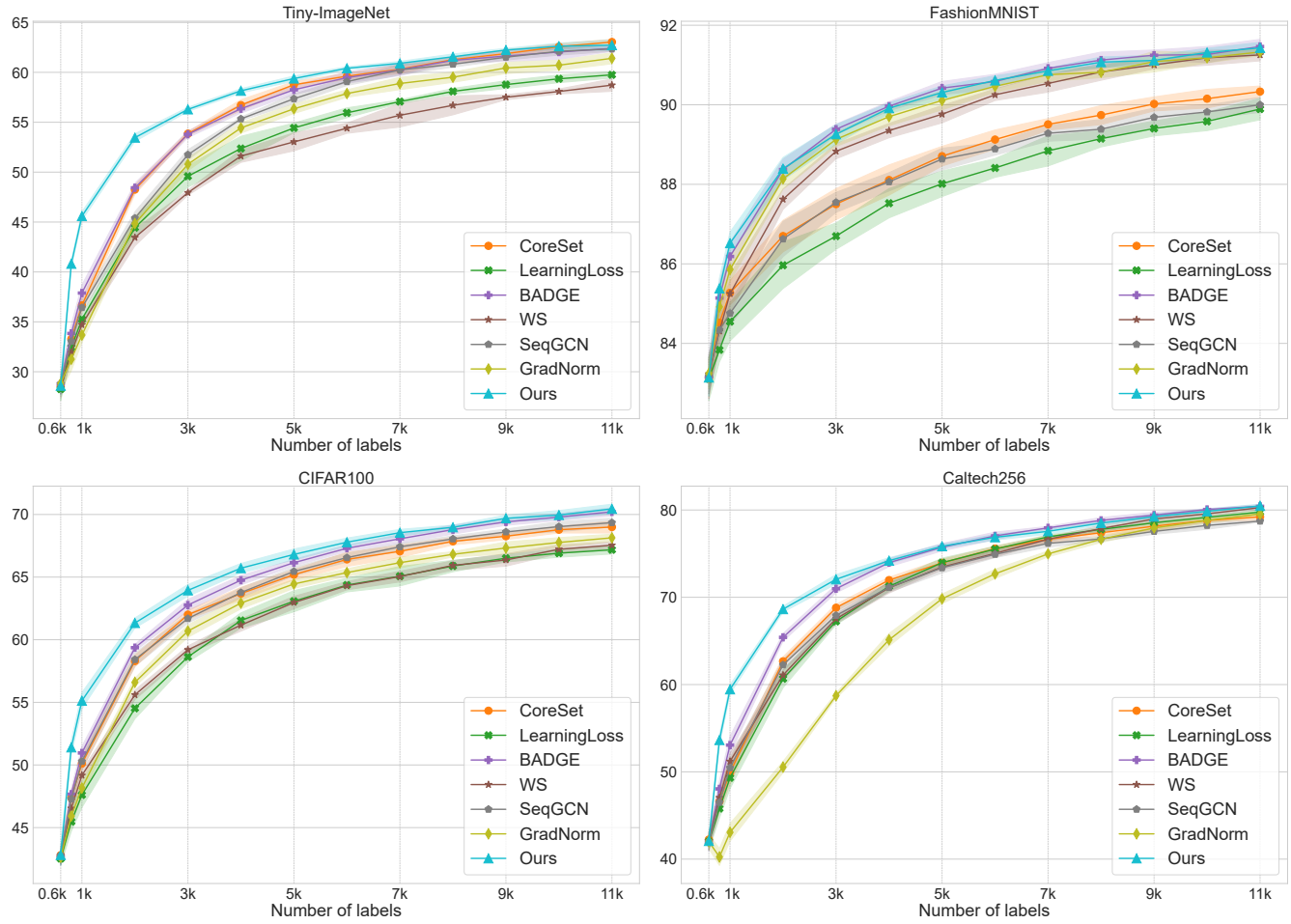

Figure 1: Mean accuracy (in %) across ten repeated experiments for different active learning algorithms including *CoreSet* (Sener and Savarese 2018), *LearningLoss* (Yoo and Kweon 2019), *BADGE* (Ash et al. 2020), *SeqGCN* (Caramalau et al. 2021), *GradNorm* (Wang et al. 2022), weight decay scheduling (*WS*) (Yun et al. 2020), and our algorithm, with random network initializations. The widths of the shaded regions represent twice the standard deviations.

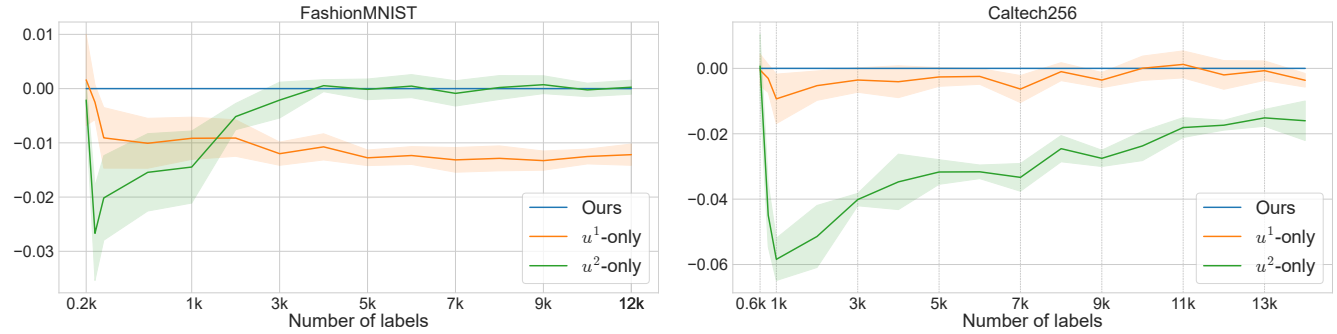

Figure 2: Performance comparison of two variants of our final algorithm: u^1-*only* and u^2-*only*, using solely the u^1 and u^2 utilities, respectively. The y-axis represents the accuracy difference compared to our final algorithm. Negative differences indicate superior performance of the final version. Our final algorithm achieves a favorable balance between maximizing influence and reducing uncertainty by leveraging the complementary strengths of the u^1 and u^2 utilities.

4 Experiments

Settings: We evaluated the performance of our algorithm using four benchmark datasets: The Tiny-ImageNet dataset (Le and Yang 2015), CIFAR100 (Krizhevsky 2009), FashionMNIST (Xiao et al. 2017), and Caltech256 (Griffin et al. 2007) datasets. For comparison, we also performed

experiments with Sener and Savarese (2018)'s core-set approach (*CoreSet*), Yoo and Kweon (2019)'s learning loss (*LearningLoss*), Caramalau et al. (2021)'s sequential GCN-based algorithm (*SeqGCN*), Ash et al. (2020)'s batch AL by diverse gradient embeddings (*BADGE*), Wang et al. (2022)'s gradient norm-based approach (*GradNorm*), and Yun et al. (2020)'s weight decay scheduling scheme (*WS*). In the accompanying supplementary document (An et al. 2023), we also present the results of Sinha et al. (2019)'s variational adversarial active learning (*VAAL*) and the task-aware VAAL-extension proposed by Kim et al. (2021) (*TA-VAAL*). Throughout the experiments, we initiated the process by randomly selecting 600 images and labeling them, which were then used to train the baseline learner. Subsequently, the AL algorithms augmented the labeled set until it reached the final budget of B =11,000. To evaluate the label acquisition performance during the early stages of learning, the learners were assessed at 600, 800, and 1,000 labels, followed by evaluations at every 1,000 additional labels (I=1,000). These experiments encompassed a range of labeling budgets $B = \{600, 800, 1000, \ldots, 11,000\}$.

For the baseline learner of the AL algorithms, we initially evaluated ResNet18 (He et al. 2016), ResNet101 (He et al. 2016), and VGG16 (Simonyan and Zisserman 2015), all combined with fully connected (FC) layers matching the number of classes per dataset. Among them, we selected a ResNet101 pre-trained on ImageNet; Combining the *pool5* layer of ResNet101 with three FC layers consistently outperformed the other networks. Our learners were trained using stochastic gradient descent with an initial learning rate of 0.01. The learning rate was reduced to 10% for every 10 epochs. The mini-batch size and the total number of epochs were fixed at 30 and 100, respectively. For the GP surrogate $\hat{f}$, we used the *pool5* layer outputs of ResNet101 as inputs $\mathbf{x}$. All experiments were repeated ten times with random initializations and the results were averaged.

Results: Figure 1 presents a summary of the results. While *CoreSet* is effective in identifying diverse data points by analyzing the overall distribution of data, its performance tends to decline in later stages of learning when the baseline learner f provides more reliable uncertainty estimates. A comparable behavior was demonstrated by *VAAL* (in the supplementary document (An et al. 2023)). This is because diversity-based methods do not directly leverage f's predictions. On the other hand, *WS* and *LearningLoss* use this task-specific information to achieve higher accuracies compared to *CoreSet* in later stages of Caltech256.

The performance of the different algorithms exhibited notable variations across the datasets. FashionMNIST, consisting of only 10 classes, demonstrated that even with a limited number of labeled samples, the baseline learners produced highly accurate uncertainty predictions. Consequently, uncertainty-based methods, *LearningLoss* and *WS* showed superior performance on these datasets. Conversely, CIFAR100, Caltech256, Tiny-ImageNet, with a larger number of classes, posed challenges as the class predictions and uncertainty estimates of the learner f became unreliable, even with a larger number of labeled samples. In such cases, diversity-based method, namely *CoreSet* demonstrated higher accuracies.

By combining diversity and uncertainty, *BADGE* achieved significantly higher accuracies than methods relying solely on either diversity or uncertainty. In particular, *BADGE* attained the best performance among the baseline methods in the early stages of FashionMNIST learning. By incorporating these two AL modes into a single Gaussian process model, capturing the continuous learning behavior of f, our algorithm exhibited further significant improvements on CIFAR100, Caltech256, and Tiny ImageNet. For FashionMNIST, our algorithm's results were on par with the respective best-performing algorithms, namely *WS* and *BADGE*. The overall accuracies achieved by all algorithms in our experiments were considerably higher than the results reported in previous works (Ash et al. 2020; Sener and Savarese 2018; Yoo and Kweon 2019; Sinha et al. 2019; Kim et al. 2021), primarily due to the use of stronger baseline learners.

Contributions of Influence and Uncertainty: We evaluated two variations of our final algorithm, using only the u^1 utility (u^1-*only*) and the u^2 utility (u^2-*only*). Figure 2 shows the results. Our influence utility u^1 demonstrated greater effectiveness on CIFAR100 and in the early learning stages of FashionMNIST, where the predictions of the learner were less accurate. However, as the baseline learner provided more reliable confidence estimates in the later learning stages of FashionMNIST, its performance advantage over u^2-*only* diminished. Conversely, our uncertainty-based utility u^2 exhibited the opposite behavior, performing better in later learning stages of FashionMNIST. By leveraging their complementary strengths, thereby trading influence and uncertainty, our final algorithm consistently outperformed u^1-*only* and u^2-*only*.

5 Conclusions

We have introduced a novel active learning algorithm that leverages a Gaussian process (GP) model as a surrogate for the baseline neural network learner, effectively identifying influential and difficult data points. By using the well-established Bayesian framework, our algorithm offers a rigorous approach to maximizing the information gain at each stage of the active learning process. To identify difficult points, an efficient GP surrogate is instantly updated each time a single label is provided with each new labeled instance. This allows us to faithfully simulate the continuous learning behavior of the baseline learner without the need for retraining. Consequently, we can avoid introducing additional mechanisms to promote label diversity, which often necessitate tuning separate hyperparameters.

Our GP surrogate $\hat{f}$ may deviate from f. Empirically, we have observed that $\hat{f}$ faithfully captures the behavior of f due to the use of the product kernel k (Eq. 1). Nonetheless, a theoretical analysis of the quality of $\hat{f}$ as a surrogate for f and its impact on the resulting active learning performance would provide deeper insights into the utility of our algorithm. Future work should explore this.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant (No. 2021R1A2C2012195) and the Institute of Information & communications Technology Planning & Evaluation (IITP) grants (No.2019-0-01906, Artificial Intelligence Graduate School Program, POSTECH, and 2020–0–01336, Artificial Intelligence Graduate School Program, UNIST), all funded by the Korea government (MSIT).

References

Aggarwal, C. C.; Kong, X.; Gu, Q.; Han, J.; and Yu, P. S. 2015. Active learning: a survey. In Aggarwal, C. C., ed., *Data Classification Algorithms and Applications*, 571–605. CRC Press.

An, Y.; Park, S.; and Kim, K. I. 2023. Active learning guided by efficient surrogate learners. In *arXiv:2301.02761*.

Aodha, O. M.; Campbell, N. D. F.; Kautz, J.; and Brostow, G. J. 2014. Hierarchical subquery evaluation for active learning on a graph. In *CVPR*, 564–571.

Ash, J. T.; Goel, S.; Krishnamurthy, A.; and Kakade, S. M. 2021. Gone fishing: neural active Learning with Fisher embeddings. In *NeurIPS*.

Ash, J. T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; and Agarwal, A. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *ICLR*.

Caramalau, R.; Bhattarai, B.; and Kim, T.-K. 2021. Sequential graph convolutional network for active learning. In *CVPR*, 9583–9592.

Coleman, C.; Yeh, C.; Mussmann, S.; Mirzasoleiman, B.; Bailis, P.; Liang, P.; Leskovec, J.; and Zaharia, M. 2020. Selection via proxy: efficient data selection for deep learning. In *ICLR*.

Elhamifar, E.; Sapiro, G.; Yang, A.; and Sastry, S. S. 2013. A convex optimization framework for active learning. In *ICCV*, 4321–4328.

Freytag, A.; Rodner, E.; and Denzler, J. 2014. Selecting influential examples: active learning with expected model output changes. In *ECCV*, 562–577.

Geifman, Y.; and El-Yaniv, R. 2017. Deep active learning over the long tail. In *arXiv:1711.00941*.

Gissin, D.; and Shalev-Shwartz, S. 2019. Discriminative active learning. In *arXiv:1907.06347*.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset. Technical report, California Institute of Technology.

Guo, Y. 2010. Active instance sampling via matrix partition. In *NIPS*, 802–810.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Houlsby, N.; Huszár, F.; Ghahramani, Z.; and Lengyel, M. 2011. Bayesian active learning for classification and preference learning. In *arXiv:1112.5745*.

Kim, K.; Park, D.; Kim, K. I.; and Chun, S. Y. 2021. Task-aware variational adversarial active learning. In *CVPR*, 8166–8175.

Kirsch, A.; van Amersfoort, J.; and Gal, Y. 2019. BatchBALD: efficient and diverse batch acquisition for deep Bayesian active learning. In *NeurIPS*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Le, Y.; and Yang, X. 2015. Tiny ImageNet visual recognition challenge. Technical Report CS231N Course, Stanford University.

Nguyen, H. T.; and Smeulder, A. 2004. Active learning using pre-clustering. In *ICML*, 623–630.

Rasmussen, C. E.; and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Chen, X.; and Wang, X. 2020. A survey of deep active learning. In *arXiv:2009.00236*.

Schott, J. R. 2016. *Matrix Analysis for Statistics*. New Jersey: Wiley, 3rd edition.

Sener, O.; and Savarese, S. 2018. Active learning for convolutional neural networks: a core-set approach. In *ICLR*.

Settles, B. 2009. Active learning literature survey. Technical Report Computer Sciences Technical Report 1648, University of Wisconsin-Madison.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*, arXiv:1409.1556.

Sinha, S.; Ebrahimi, S.; and Darrell, T. 2019. Variational adversarial active learning. In *ICCV*, 5972–5981.

Snelson, E.; and Ghahramani, Z. 2006. Sparse Gaussian processes using pseudo-inputs. In *NIPS*.

Sollich, P.; and Williams, C. K. I. 2005. Using the equivalent kernel to understand Gaussian process regression. In *NIPS*, 1313–1320.

Tong, S.; and Koller, D. 2001. Support vector machine active learning with applications to text classification. *JMLR*, 2: 45–66.

Tran, T.; Do, T.-T.; Reid, I.; and Carneiro, G. 2019. Bayesian Generative Active Deep Learning. In *ICML*, 6295–6304.

Vijayanarasimhan, S.; and Kapoor, A. 2010. Visual recognition and detection under bounded computational resources. In *CVPR*, 562–577.

Wang, T.; Li, X.; Yang, P.; Hu, G.; Zeng, X.; Huang, S.; Xu, C.-Z.; and Xu, M. 2022. Boosting active learning via improving test performance. In *AAAI*, 8566–8574.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. FashionMNIST: a novel image dataset for benchmarking machine learning algorithms. In *arXiv:1708.07747*.

Yoo, D.; and Kweon, I. S. 2019. Learning loss for active Learning. In *CVPR*, 93–102.

Yun, J.; Kim, B.; and Kim, J. 2020. Weight decay scheduling and knowledge distillation for active learning. In *ECCV*, 431–447.

Zhang, B.; Li, L.; Yang, S.; Wang, S.; Zha, Z.-J.; and Huang, Q. 2020. State-relabeling adversarial active learning. In *CVPR*, 8756–8765.