Training-Free Neural Active Learning with Initialization-Robustness Guarantees

Apivich Hemachandra [1] **Zhongxiang Dai** [1] **Jasraj Singh** [2] **See-Kiong Ng** [1] **Bryan Kian Hsiang Low** [1]

Abstract

Existing neural active learning algorithms have aimed to optimize the predictive performance of neural networks (NNs) by selecting data for labelling. However, other than a good predictive performance, being robust against random parameter initializations is also a crucial requirement in safety-critical applications. To this end, we introduce our *expected variance with Gaussian processes* (EV-GP) criterion for neural active learning, which is theoretically guaranteed to select data points which lead to trained NNs with both (a) good predictive performances and (b) initialization robustness. Importantly, our EV-GP criterion is training-free, i.e., it does not require any training of the NN during data selection, which makes it computationally efficient. We empirically demonstrate that our EV-GP criterion is highly correlated with both initialization robustness and generalization performance, and show that it consistently outperforms baseline methods in terms of both desiderata, especially in situations with limited initial data or large batch sizes.

1. Introduction

Deep neural networks (NNs) have recently achieved impressive performances in various applications thanks to the availability of massive amount of labelled data. However, in some applications, data labelling is so costly that obtaining large datasets is infeasible. In this regard, a number of works have used the method of *neural active learning* to select a small number of data points to be labelled and subsequently be used for the training of the NNs (Ren et al., 2021; Sener & Savarese, 2018; Kirsch et al., 2019; Ash et al., 2020).

When selecting the data for labelling, existing neural active learning methods have only aimed to optimize the predictive performance (e.g., generalization performance) of the trained NNs. However, the output from a neural network is heavily dependent on its training process, especially on its *initialization* (Cyr et al., 2019; Lee et al., 2020). Therefore in some safety-critical applications, *having small variability in the trained NNs with respect to model initialization* is also a crucial requirement. For example, given a training set of patient data in a healthcare application (e.g., disease diagnosis), if the NNs trained using different random initializations have drastically disparate predictions, the resulting NNs would be too unreliable to be used for clinical diagnosis due to the high-stakes nature of the application (Esteva et al., 2019). As a simple illustration, Fig. 1 shows that different training sets can indeed result in distinct sensitivities of the trained NNs to model initialization, and hence a badly constructed training set (e.g., the rightmost figure in Fig. 1) can cause the trained NN to be highly variable (and hence untrustworthy) w.r.t. the model initialization. Therefore, in these important applications, it is paramount for a neural active learning algorithm to select data points which lead to trained NNs with not only (a) good predictive performances but also (b) *initialization robustness*.

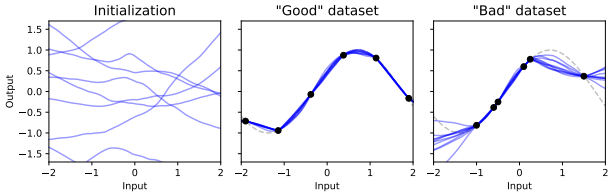

Figure 1. Neural networks initialized randomly with different seeds (left), after training, can give consistent outputs (center) or highly varying outputs (right) depending on the training data.

To this end, we introduce our *Expected Variance with Gaussian Processes* (EV-GP) criterion, which is theoretically shown to select data points which satisfy both criteria. Specifically, we firstly leverage the theory of *neural tangent kernel* (NTK) (Jacot et al., 2018) and overparameterized NNs (Lee et al., 2020) to characterize the predictive distribution of trained NNs (Sec. 3.1) with respect to random model initialization. This allows us to measure the robustness of the trained NNs against model initialization via the output variance, i.e., a smaller output variance indicates a more initialization-robust NN. Next, we introduce an efficiently computable approximation of the output variance, and de-

[1]Department of Computer Science, National University of Singapore, Republic of Singapore [2]School of Computer Science and Engineering, Nanyang Technological University, Republic of Singapore. Correspondence to: Zhongxiang Dai <dzx@nus.edu.sg>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

rive a theoretical guarantee on the approximation quality (Sec. 3.1). Subsequently, we show that this approximate output variance, which is an indicator of (the inverse of) initialization robustness, is also, interestingly, an upper bound on the generalization error of the trained NNs (Sec. 3.2).

As a result, our EV-GP active learning criterion (Sec. 3.3), which is based on the minimization of the above-mentioned approximate output variance, is able to select data points which lead to both (a) good predictive performances (i.e., small generalization errors) and (b) initialization robustness. Moreover, our EV-GP criterion also has the additional advantages of *computational efficiency* and *generality*. Firstly, during data selection, the computation of our EV-GP criterion *does not require training of the NN*, which is usually computationally costly. Therefore, our EV-GP criterion is computationally efficient and can be used to select all data points in a small number of batches or even a single batch. Secondly, our EV-GP criterion does not impose any requirement on the training process of the NN, and is hence applicable to generic NNs rather than requiring specific types of models such as Bayesian NNs. Furthermore, we introduce a theoretically grounded method for model selection which is able to automatically optimize the architecture of the NN based on the selected data points (Sec. 4).

We empirically show that our EV-GP criterion can accurately characterize the output variance (w.r.t. the random model initialization) and hence the initialization robustness of trained NNs (Secs. 5.1 and 5.2). Furthermore, we use extensive real-world regression (Sec. 5.2) and classification (Sec. 5.3) experiments to show that our EV-GP criterion outperforms existing baselines in terms of both initialization robustness and predictive performances. Of note, the performance advantage of our EV-GP criterion is particularly pronounced when there is limited labelled data or the batch size is large (Sec. 5.3). Lastly, we also empirically verify that our model selection method (Sec. 4) is able to further improve the performance of our EV-GP criterion (Sec. 5.4).

2. Problem Setup and Background

In our problem setup, consistent with previous works on active learning, we have access to an unlabelled pool of input data $\mathcal{X}_U$ whose corresponding labels are expensive to obtain. Also, suppose we have a set of testing input data $\mathcal{X}_T$ (whose corresponding labels are unknown), which we want our NN to eventually make predictions on. In practice, when a set of testing input is not available, we can simply let $\mathcal{X}_T = \mathcal{X}_U$, which we have done in our experiments. Additionally, we are given a set $\mathcal{L}_0 = (\mathcal{X}_0, \mathbf{y}_0)$ of already-labelled data (which may be empty), and the algorithm proceeds to construct a set $\mathcal{L} = (\mathcal{X}_\mathcal{L}, \mathbf{y}_\mathcal{L})$ where $\mathcal{X}_\mathcal{L} \setminus \mathcal{X}_0 \subset \mathcal{X}_U$ and $|\mathcal{L} \setminus \mathcal{L}_0| \leq k$. The parameter k can be considered the budget for active learning, which is a limit on the number of queries to the oracle. To construct $\mathcal{L}$, in each round of active learning, the algorithm selects a set of b unlabelled input data points $\mathcal{X}_b \subset \mathcal{X}_U$ and submits them to the oracle to obtain their corresponding labels $\mathbf{y}_b$. It is desirable to have minimal training of the NN between different rounds of querying in order to avoid incurring excessive computational costs and long delays between each rounds.

Once the labelled training set $\mathcal{L}$ is obtained after the active learning algorithm, we use it as the training set $\mathcal{D} = \mathcal{L}$ to train an NN, $f(\cdot;\theta)$, where θ denotes its parameters. The model parameters are firstly initialized, $\theta_0 \sim \text{init}(\theta)$, and following the common practice (Glorot & Bengio, 2010), we assume that each parameter of θ_0 is independently drawn from a Gaussian distribution with zero mean and a known variance. After initialization, the model is trained with gradient-based optimization to minimize the regularized mean-squared error (MSE) loss:

$$\ell(\mathcal{D};\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \frac{1}{2}\|f(x;\theta) - y\|^2 + \frac{\lambda}{2}\|\theta\|^2 \,. \quad (1)$$

The loss function comprises of a mean-squared error term (first term) and a regularisation term (second term), in which λ controls their trade-off. Note that the regularized MSE loss (1) is only required in our theoretical analysis and in practice, other loss functions such as the cross-entropy loss can also be used (Sec. 5.3). The NN model is assumed to be trained till convergence, resulting in the parameters $\theta_\infty = \text{train}(\theta_0)$ where $\text{train}(\cdot)$ denotes the function of the training process. As discussed in Sec. 1, we would like the final model predictions $f(x;\text{train}(\theta_0))$ to achieve both small generalization errors and low output variances with respect to the randomness of θ_0.

2.1. Neural Tangent Kernels

When training a NN using a training set $\mathcal{D} = (\mathcal{X}, \mathbf{y})$ with gradient descent (GD): $\theta_{t+1} \leftarrow \theta_t - \eta \cdot \nabla_\theta \ell(\mathcal{D};\theta_t)$ where η is the learning rate, it has been shown (Lee et al., 2020) that as long as η is small enough, the training can be approximated by continuous GD. As a result, the change in the predictive output $f(\mathcal{X};\theta_t)$ over time can be expressed as:

$$\frac{\partial f(\mathcal{X};\theta_t)}{\partial t} = -\eta \underbrace{\nabla_\theta f(\mathcal{X};\theta_t)\nabla_\theta f(\mathcal{X};\theta_t)^\top}_{\triangleq \hat{\Theta}_t(\mathcal{X},\mathcal{X})} \nabla_f \ell(\mathcal{D};\theta) \,. \quad (2)$$

The term $\hat{\Theta}_t(\mathcal{X},\mathcal{X}') \triangleq \nabla_\theta f(\mathcal{X};\theta_t)\nabla_\theta f^\top(\mathcal{X}';\theta_t)$ is referred to as the (empirical) *neural tangent kernel* (NTK) (Jacot et al., 2018; Arora et al., 2019b). While kernels are defined over a tuple of elements from the input space, we will overload the notation for all kernels and use $\Theta(\mathcal{X},\mathcal{X}') = (\Theta(x,x'))_{x\in\mathcal{X},x'\in\mathcal{X}'}$ to represent the matrix constructed using values of the kernel. We will sometimes use the shorthand notation $\Theta_\mathcal{X}$ to represent $\Theta(\mathcal{X},\mathcal{X})$ and $\Theta_{\mathcal{X}'\mathcal{X}}$ to represent $\Theta(\mathcal{X}',\mathcal{X})$ when the context is clear, which is consistent with previous works on NTK (He et al., 2020).

The work of Lee et al. (2020) has shown that when the width of the NN approaches infinity, the output of the NN can be approximated by a linear model:

$$f(x;\theta) \approx f(x;\theta_0) + \langle \nabla_\theta f(x;\theta_0), \theta - \theta_0 \rangle \ , \qquad (3)$$

and the empirical NTK $\hat{\Theta}_t(\cdot,\cdot)$ stays constant throughout training and approaches a deterministic kernel $\Theta(\cdot,\cdot)$ regardless of the initialization (Lee et al., 2020). Using the linear approximation (3), Lee et al. (2020) have shown that if a randomly initialized NN is trained on data $(\mathcal{X}, \mathbf{y})$ with the mean-squared error loss ((1) with $\lambda = 0$) till convergence, then the predictions of the converged model on a testing set $\mathcal{X}_T$ follow a normal distribution: $f(\mathcal{X}_T) \sim \mathcal{N}(\mu(\mathcal{X}_T|\mathcal{X}, \mathbf{y}), \Sigma_{\text{NN}}(\mathcal{X}_T|\mathcal{X}))$, where the output mean is

$$\mu_{\text{NN}}(\mathcal{X}_T|\mathcal{X}, \mathbf{y}) = \Theta_{\mathcal{X}_T\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathbf{y} \ , \qquad (4)$$

and the output covariance is

$$\Sigma_{\text{NN}}(\mathcal{X}_T|\mathcal{X}) = \mathcal{K}_{\mathcal{X}_T} + \Theta_{\mathcal{X}_T\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}\mathcal{X}_T} - \left(\Theta_{\mathcal{X}_T\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}\mathcal{X}_T} + \mathcal{K}_{\mathcal{X}_T\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}\mathcal{X}_T}\right) . \qquad (5)$$

When the testing set $\mathcal{X}_T$ consists of only a single point x, we use $\sigma^2_{\text{NN}}(x|\mathcal{X}) = \Sigma_{\text{NN}}(\mathcal{X}_T|\mathcal{X})$ to denote the predictive variance at x. The kernel $\mathcal{K}$ in (5), which is defined as $\mathcal{K}(x, x') = \mathbb{E}_{\theta_0 \sim \text{init}(\theta)}[f(x;\theta_0) \cdot f(x';\theta_0)]$, is the covariance of the NN output with respect to the random initialization. Similar to Θ above, we have used the shorthand notations $\mathcal{K}_{\mathcal{X}}$ and $\mathcal{K}_{\mathcal{X}'\mathcal{X}}$ to represent $\mathcal{K}(\mathcal{X}, \mathcal{X})$ and $\mathcal{K}(\mathcal{X}', \mathcal{X})$, respectively. We will also use $\mu_{\text{NN},f}$ and $\sigma^2_{\text{NN},f}$ to represent the predictive mean and variance calculated using a particular model architecture f.

2.2. Gaussian Processes

To derive our active learning criterion (Sec. 3), we will make use of tools from the literature of *Gaussian processes* (GPs) (Rasmussen & Williams, 2006). A GP is fully specified by a prior mean function $\mu(x)$ which is usually assumed to be $\mu(x) = 0 \ \forall x$, and a covariance function $K(x, x')$ (also called kernel function). Given some data $(\mathcal{X}, \mathbf{y})$, the GP posterior predictive distribution of the outputs $\mathbf{y}_T$ at a testing set $\mathcal{X}_T$ (in the noiseless setting) is given by $\mathcal{N}(K_{\mathcal{X}_T\mathcal{X}}K_{\mathcal{X}}^{-1}\mathbf{y}, K_{\mathcal{X}_T} - K_{\mathcal{X}_T\mathcal{X}}K_{\mathcal{X}}^{-1}K_{\mathcal{X}\mathcal{X}_T})$. The principled uncertainty measures provided by GPs have been used for data selection in active learning (Krause & Guestrin, 2007; Krause et al., 2008; Hoang et al., 2014a;b; Ling et al., 2016; Zhang et al., 2016; Nguyen et al., 2021c; Xu et al., 2023). For example, Krause & Guestrin (2007) have used the mutual information between the selected data and the unlabelled data, calculated using GPs, as the active learning criterion.

3. Initialization-Robust Active Learning

In this section, we firstly introduce a computationally efficient approximation of the output variance of an NN w.r.t. random initializations, and derive a theoretical guarantee on the approximation quality (Sec. 3.1). Next, we show that the approximate output variance from Sec. 3.1, is also, interestingly, an upper bound on the generalization error of the NN (Sec. 3.2). Finally, we use the approximate output variance to design our active learning criterion (Sec. 3.3).

3.1. Approximation of Output Variance of Trained Neural Network

In the infinite-width regime, when an NN with initial parameters θ_0 is trained till convergence to yield parameters $\text{train}(\theta_0)$, the output of the NN on a test set $\mathcal{X}_T$ follows a normal distribution with mean $\mu_{\text{NN}}(\mathcal{X}_T|\mathcal{X}, \mathbf{y})$ ((4)) and covariance $\Sigma_{\text{NN}}(\mathcal{X}_T|\mathcal{X})$ (5) (Lee et al., 2020). The randomness in the output distribution results from the random initializations θ_0. Given this, we see that Σ_{NN} can serve as a natural and principled measure of initialization robustness.

However, Σ_{NN} presents a significant computational challenge because it requires computing two different kernels (i.e., Θ and $\mathcal{K}$) and a number of matrix inversion and multiplication operations. Therefore, by drawing inspiration from GPs (Rasmussen & Williams, 2006), we introduce an approximation of Σ_{NN} which is both computationally efficient (Sec. 3.1.1) and equipped with a theoretical guarantee on the approximation quality (Sec. 3.1.2).

3.1.1. Computational Efficiency

Given some data $(\mathcal{X}, \mathbf{y})$, performing GP regression with NTK as the covariance function (which we refer to as NTKGP following He et al. (2020)) leads to the output distribution (on a testing set $\mathcal{X}_T$) of $f(\mathcal{X}_T) \sim \mathcal{N}(\mu_{\text{NTKGP}}(\mathcal{X}_T|\mathcal{X}, \mathbf{y}), \Sigma_{\text{NTKGP}}(\mathcal{X}_T|\mathcal{X}))$, where $\mu_{\text{NTKGP}} = \mu_{\text{NN}}$ (4) and

$$\Sigma_{\text{NTKGP}}(\mathcal{X}_T|\mathcal{X}) = \Theta_{\mathcal{X}_T} - \Theta_{\mathcal{X}_T\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}\mathcal{X}_T} \ . \qquad (6)$$

When $\mathcal{X}_T$ contains a single point x, we use $\sigma^2_{\text{NTKGP}}(x|\mathcal{X}) = \Sigma_{\text{NTKGP}}(\mathcal{X}_T|\mathcal{X})$ to denote the output variance at x.

Compared with Σ_{NN}, the covariance function Σ_{NTKGP} is more efficient to compute because (a) it only requires computing one (instead of two) kernel Θ which can also be easily approximated using the inner product of the parameter gradients at initialization (Sec. 2.1), and (b) the posterior distributions of GPs are well-studied, allowing us to adopt existing tools to further reduce the computational cost of Σ_{NTKGP}. Even though Σ_{NTKGP} incurs a cost of $\mathcal{O}(n^3)$ with n queried data points due to inversion of an n-by-n matrix, we are able to use results from linear algebra to perform low-rank updates to the GP covariance to reduce the running time to $\mathcal{O}(n^2)$. Furthermore, we can adopt techniques from the abundant literature of sparse GPs to significantly reduce the dependency on n (Quiñonero-Candela & Rasmussen, 2005; Hoang et al., 2015). We discuss the various

approximation methods for Σ_{NTKGP} in App. C.

3.1.2. Guaranteed Approximation Quality

To provide a theoretical justification for our approximation Σ_{NTKGP} (6), we need to theoretically bound the difference between Σ_{NTKGP} and Σ_{NN}. The work of He et al. (2020) has shown that $\Sigma_{\text{NN}}(\mathcal{X}'|\mathcal{X}) \preceq \Sigma_{\text{NTKGP}}(\mathcal{X}'|\mathcal{X})$. In other words, for a single test point x, we have that $\sigma^2_{\text{NN}}(x|\mathcal{X}) \leq \sigma^2_{\text{NTKGP}}(x|\mathcal{X})$, which suggests that the NTKGP approximation (6) is an overestimation of the true output variance (5). However, this result is not enough for guaranteeing a small approximation error. Therefore, we prove here a stronger connection between $\sigma^2_{\text{NN}}(x|\mathcal{X})$ and $\sigma^2_{\text{NTKGP}}(x|\mathcal{X})$:

Theorem 3.1 (Informal)**.** *Let $f(\cdot;\theta)$ be an infinite-width NN with ReLU activation and $L \geq 2$ hidden layers whose NTK satisfies $|\Theta(x,x')| \leq B$ for all $x, x' \in \mathcal{X}$. Then, there exist some constants $\alpha > 0$ and $\beta = \mathcal{O}\left(\text{poly}(|\mathcal{X}|, B, L, \lambda_{\min}(\Theta_{\mathcal{X}})^{-1})\right)$ such that*

$$\left|\sigma^2_{\text{NN}}(x|\mathcal{X}) - \alpha \cdot \sigma^2_{\text{NTKGP}}(x|\mathcal{X})\right| \leq \beta \,.$$

Theorem 3.1 shows that $\sigma^2_{\text{NN}}(x|\mathcal{X})$ and $\alpha \cdot \sigma^2_{\text{NTKGP}}(x|\mathcal{X})$ have a bounded difference and are hence expected to behave similarly. Of note, when using the active learning criterion based on σ^2_{NTKGP} (Sec. 3.3) to select data points to query, what affects the selected points is only the *relative* ranking of the values of the criterion (at different inputs in the unlabelled pool), therefore, the presence of the constant $\alpha > 0$ does not affect our active learning algorithm. The approximation error in Theorem 3.1 depends on a number of factors including the model architecture (which affects the difference between Θ and $\mathcal{K}$), the number of points in $\mathcal{X}$ (due to the accumulation of the approximation errors with more data points), and the eigenvalues of $\Theta_{\mathcal{X}}$ (which affects how "well-behaved" the matrix $\Theta_{\mathcal{X}}$ is).

As a brief sketch of the proof, firstly, it can be verified that if $\mathcal{K}(x,x') = \alpha \cdot \Theta(x,x')$, then $\sigma^2_{\text{NN}}(x|\mathcal{X}) = \alpha \cdot \sigma^2_{\text{NTKGP}}(x|\mathcal{X})$. However, in general, $\mathcal{K}(x,x') \neq \alpha \cdot \Theta(x,x')$. Instead, we have managed to show that the ratio between $\mathcal{K}(x,x')$ and $\Theta(x,x')$ is bounded, i.e. there exists some constants $a_- > 0$ and $a_+ > 0$ (which depend on L) such that

$$a_- \leq \mathcal{K}(x,x')/\Theta(x,x') \leq a_+ \,. \tag{7}$$

As a result, (7) allows us to bound $\left|\sigma^2_{\text{NN}}(x|\mathcal{X}) - \alpha \cdot \sigma^2_{\text{NTKGP}}(x|\mathcal{X})\right|$. The complete proof is presented in App. A. We will also provide empirical justifications for Theorem 3.1 in Sec. 5.1 by showing that σ^2_{NTKGP} is indeed highly correlated with the empirical output variance of the NN resulting from different model initializations, and is hence a reliable indicator of initialization robustness.

3.2. Connection with Generalization Error

In this section, we show that the approximate output variance σ^2_{NTKGP} (6) is also an upper bound on the generalization error of the trained NN and hence a good indicator of its predictive performance. To analyze the performance of the trained neural network, we make the following assumption about the groundtruth function f^* in a manner similar to Vakili et al. (2021b).

Assumption 3.2. Assume that the groundtruth function $f^* \in \mathcal{H}_\Theta$. Specifically, f^* lies in the reproducing kernel Hilbert space (RKHS) of the NTK Θ, or equivalently, its RKHS norm is such that $\|f^*\|_{\mathcal{H}_\Theta} \leq B$ for some $B \in \mathbb{R}_{\geq 0}$. Further assume that the function observation at any input x_i is given by $y_i = f^*(x_i) + \xi_i$, in which every ξ_i is i.i.d. observation noise drawn from an R sub-Gaussian distribution: $\mathbb{E}[\exp(\eta \xi_i)] \leq \exp(\eta^2 R^2/2)$ for all $\eta \in \mathbb{R}$.

Both assumptions in Assumption 3.2 are commonly made in the analysis of kernelized and neural bandits (Chowdhury & Gopalan, 2017; Kassraie & Krause, 2022). They allow us to show the following theoretical guarantee on the generalization error:

Theorem 3.3 (Informal)**.** *Suppose we train an infinitely wide NN $f(\cdot;\theta)$ with training dataset $(\mathcal{X}, \mathbf{y})$ on mean-squared error loss function using gradient descent until convergence. Then, there exists a constant $\zeta = \mathcal{O}\big(\text{poly}(B, R)\big)$ such that for any $x \in \mathcal{X}$, with high probability over the random observation noise $\boldsymbol{\varepsilon}$ and network initialization θ_0,*

$$\left|f^*(x) - f(x; \text{train}(\theta_0))\right| \leq \zeta \cdot \sigma_{\text{NTKGP}}(x|\mathcal{X}). \tag{8}$$

Theorem 3.3 shows that the generalization error of a trained NN through gradient descent is proportional to σ_{NTKGP} (6), where the constant of proportionality ζ is *independent* of x and $\mathcal{X}$. As a result, minimizing σ_{NTKGP} will not only (a) decrease the output variance (Sec. 3.1) and hence *improve initialization robustness*, but also (b) reduce the generalization error and hence *enhance the predictive performance*. The degree of correlation between σ_{NTKGP} and the generalization performance, represented by the constant ζ, depends on the parameters B and R, such that the easier the function f^* is to learn (i.e., a smaller B) or the less noisy the observations are (i.e., a smaller R), the better the degree of correlation. Theorem 3.3 is also consistent with the empirically observed characteristics of over-parameterized NNs, because NN models with lower variance are also observed to have higher predictive accuracy (Neal et al., 2018). Theorem 3.3 will be stated formally and proved in App. B.

3.3. Active Learning Criterion

Since we have shown that minimizing $\sigma_{\text{NTKGP}}(x|\mathcal{X})$ can improve both *initialization robustness* (Sec. 3.1) and *generaliztion performance* (Sec. 3.2), we design our active learning criterion based on the minimization of $\sigma_{\text{NTKGP}}(x|\mathcal{X})$ across all test input points $x \in \mathcal{X}_T$. Specifically, our EV-GP criterion encourages the selection of input data points

which result in small expected output variance $\sigma^2_{\text{NTKGP}}(x|\mathcal{X})$ across all test inputs, and we estimate the expected variance by averaging $\sigma^2_{\text{NTKGP}}(x|\mathcal{X})$ over the available test set $\mathcal{X}_T$:

$$\alpha_{\text{EV}}(\mathcal{X}) = \frac{1}{|\mathcal{X}_T|} \sum_{x \in \mathcal{X}_T} \left[\sigma^2_{\text{NTKGP}}(x|\emptyset) - \sigma^2_{\text{NTKGP}}(x|\mathcal{X})\right]. \tag{9}$$

We have added $\sigma^2_{\text{NTKGP}}(x|\emptyset)$ to the criterion so that $\alpha_{\text{EV}}(\mathcal{X}) \geq 0$ and that our criterion is to be maximized during active learning.[1] We will sometimes also use $\alpha_{\text{EV}}(\mathcal{X}; f)$ to indicate that the criterion uses the model architecture f.

Our α_{EV} criterion (9) has multiple computational benefits. *Firstly*, it is training-free, i.e., its calculation does not require any training of the NN and is hence able to sidestep significant computational costs resulting from model training. *Secondly*, it only requires calculating the variance at individual test points rather than the full covariance over the testing set. *Thirdly*, it can make use of the approximation techniques based on sparse GPs discussed in Sec. 3.1.1, for which we simply need to replace σ^2_{NTKGP} by its sparse GP counterparts in (9). *Fourthly*, it is monotone submodular, and therefore a greedy approach (i.e., select the point which gives the largest increase in the criterion in each selection round) is guaranteed to give a $(1 - \frac{1}{e})$-optimal solution (Nemhauser et al., 1978). We adopt the greedy approach in our experiments for simplicity (with some techniques for speedups which we discuss in App. E), and leave the use of other more sophisticated submodular optimization techniques to future works. Furthermore, another advantage of our α_{EV} criterion is that it is label-independent, because the calculation of σ^2_{NTKGP} (6) does not require the observations. Therefore, our criterion does not need the heuristic of pseudo-labels which is required by previous active learning algorithms (Ash et al., 2020; Mohamadi et al., 2022).

In addition to our α_{EV} criterion (9), we can also use Σ_{NTKGP} to construct alternative criteria with different characteristics. We introduce two additional criteria in App. D, which are based on, respectively, mutual information (which requires computing the full covariance matrix) and the percentile variance (which is not submodular in general).

4. Initialization-Robust Active Learning with Model Selection

A common issue with existing neural active learning algorithms is that a model architecture has to be fixed in advance and then used for the data point selection. However, in practice, especially when having no access to (labelled) data beforehand, it is infeasible to select the best model architecture prior to running the neural active learning algorithms. To this end, by leveraging the output distributions of overparameterized NNs in a similar way to Sec. 3.1, we extend our initialization-robust active learning algorithm (Sec. 3) to simultaneously select the data points to query (Sec. 3.3) *and* optimize the model architecture in a training-free manner.

Given a model architecture f and a training set $\mathcal{D} = (\mathcal{X}, \mathbf{y})$, the expected squared error of the trained model on a testing set $\mathcal{D}_T$ w.r.t. random parameter initializations is given by (proof in App. F)

$$\begin{aligned}\hat{\alpha}_{M,\mathcal{D}_T}(f;\mathcal{D}) &\triangleq \mathop{\mathbb{E}}_{\theta_0 \sim \text{init}(\theta)} \left[\ell\left(\mathcal{D}_T, \text{train}(\theta_0)\right)\right] \\ &= \frac{1}{2} \sum_{(x,y)\in\mathcal{D}_T} \Big[\underbrace{\left(y - \mu_{\text{NN},f}(x|\mathcal{X})\right)^2}_{①} + \underbrace{\sigma^2_{\text{NN},f}(x|\mathcal{X})}_{②}\Big].\end{aligned} \tag{10}$$

The first term in (10), ①, characterizes how well the trained model is able to fit the underlying function, which is related to its generalization performance. The second term, ②, represents the predictive variance of the trained model, which is an indicator of the complexity of the model. A good model architecture should be expressive enough to fit the underlying function well (i.e., have a small ①), while also not being too sensitive to the parameter initialization (i.e., have a small ②).

(10) allows us to design our model selection method based on cross validation using the queried data $\mathcal{D}$ during active learning. Specifically, we adopt the cross validation method of bootstrapping (Kohavi, 1995): we select a random subset $\mathcal{D}_T \subset \mathcal{D}$ of size κ as the testing set, and compute (10) using $\mathcal{D} \setminus \mathcal{D}_T$ as the training set. As a result, this leads to the following criterion for model selection:[2]

$$\alpha_M(f;\mathcal{D}) = -\mathop{\mathbb{E}}_{\mathcal{D}_T \subset \mathcal{D};\, |\mathcal{D}_T| = \kappa} \left[\hat{\alpha}_{M,\mathcal{D}_T}(f; \mathcal{D} \setminus \mathcal{D}_T)\right]. \tag{11}$$

In practice, the user chooses an appropriate κ and computes the empirical mean to approximate the expectation in (11). Since α_M is computed far fewer times than α_{EV} during the active learning process, it is reasonable to use σ^2_{NN} directly rather than approximating it with σ^2_{NTKGP}. As a result, in our algorithm for model selection, given a set of candidate model architectures $\mathcal{M} = \{f_1, \ldots, f_m\}$, we evaluate (11) for every architecture $f \in \mathcal{M}$ and subsequently choose the architecture which maximizes this criterion.

The full algorithm with both data and model selection, which we name EV-GP+MS, is shown in (1). The algorithm alternates between two phases. The first phase uses a fixed model architecture to greedily maximize our α_{EV} criterion (9) for data selection, and the second phase utilizes the queried data so far to select the best model architecture using our criterion in (11).

5. Experiments

When reporting the model output variance or average performance, we train an NN 50 times (regression) or 25 times

[1] This is to follow convention of other active learning methods.

[2] The negative sign is added to convert to maximization.

Algorithm 1 EV-GP+MS

Input: Initial labelled data $(\mathcal{X}_0, \mathbf{y}_0)$, unlabelled pool $\mathcal{X}_U$, candidate model architectures $\mathcal{M}$, batch size b
$(\mathcal{X}_\mathcal{L}, \mathbf{y}_\mathcal{L}) \leftarrow (\mathcal{X}_0, \mathbf{y}_0)$
Pick an initial model $f^* \in \mathcal{M}$
repeat
  // Phase 1: Data selection
  for b iterations **do**
    $x^* \leftarrow \arg\max_{x \in \mathcal{X}_U \setminus \mathcal{X}_\mathcal{L}} \alpha_{\text{EV}}(\mathcal{X}_\mathcal{L} \cup \{x\}; f^*)$
    $\mathcal{X}_\mathcal{L} \leftarrow \mathcal{X}_\mathcal{L} \cup \{x^*\}$
  end for
  Query the unlabelled points in $\mathcal{X}_\mathcal{L}$ for the labels $\mathbf{y}_\mathcal{L}$
  // Phase 2: Model selection
  $f^* \leftarrow \arg\max_{f \in \mathcal{M}} \alpha_M\big(f; (\mathcal{X}_\mathcal{L}, \mathbf{y}_\mathcal{L})\big)$
until budget exhausted
return $(\mathcal{X}_\mathcal{L}, \mathbf{y}_\mathcal{L}), f^*$

(classification) with the same architecture but with different model initializations, and then calculate the empirical mean and variance. All experiments are repeated 5 times unless stated otherwise, and their mean and standard deviations are reported. Although the theoretical properties of Σ_{NN} and Σ_{NTKGP} are applicable to infinite-width NNs, we follow the practice of previous works on NTKs (He et al., 2020; Mohamadi et al., 2022) and use finite-width NNs, because they are able to achieve good performances. In our experiments, we test our algorithm using both the theoretical NTKs (computed using the JAX-based (Bradbury et al., 2018) `Neural-Tangents` package (Novak et al., 2019)), and the empirical NTK (computed using PyTorch). We will use EV-GP-EMP to denote instances when we use the empirical NTK for our algorithm. We discuss the computation of NTKs further in App. G.2. We adopt the MSE loss (1) for regression experiments and the cross-entropy loss for classification experiments, which is consistent with previous works on NTK (Liu & Zenke, 2020; Shu et al., 2022a). We find that even though the NTK theory is developed based on MSE loss, prior works utilizing NTKs have shown that it is also effective in predicting behaviors of models trained under cross-entropy loss as well. We compare our algorithm with previous baselines which also require minimal model training between different batches and do not incur significant extra computations. These benchmarks are described further in App. G.4. In particular, we compare with random selection, K-MEANS++ (Arthur & Vassilvitskii, 2007), BADGE (Ash et al., 2020), and MLMOC (Mohamadi et al., 2022). The former two algorithms, like our algorithm, can select all the points in a single batch, whereas the latter two are not designed for such a setting but are applicable after modifications. We omit results comparing our algorithm against BATCHBALD (Kirsch et al., 2019) in the main paper due to the method requiring a Bayesian neural network, although their results have been included in H.4.2.

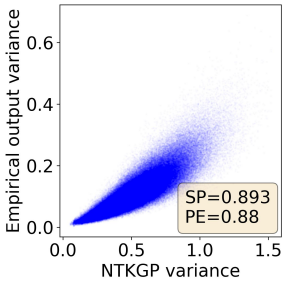

Figure 2. Correlation between our approximate output variance $\sigma^2_{\text{NTKGP}}(x|\mathcal{X})$ and the empirical NN output variance. The full description of the graph is given in App. H.1.

We have chosen to focus on the cases with low query budgets since we find this is when the neural networks tend to show a larger difference in predictive performances, and so the active learning algorithm needs to be more careful in selecting which data points to query as they will have a larger impact on the final selected models. This setting studied in our experiments is realistic since it simulates the situations where there is little or no initial labeled data and querying any data incurs a large cost. The code for the experiments can be found at `https://github.com/apivich-h/init-robust-al`. Other experimental details are deferred to App. G due to space limitation.

5.1. Correlations Between σ^2_{NTKGP} and Neural Network Output Variance

Here we study whether our approximate output variance σ^2_{NTKGP} (Sec. 3.1) can accurately reflect the output variance of NNs (w.r.t. the random initializations) and hence the initialization robustness. Fig. 2 plots the *individual* variance predicted by our NTKGP (i.e., $\sigma^2_{\text{NTKGP}}(x|\mathcal{X})$ for some x and $\mathcal{X}$) against the empirically observed output variance resulting from different random initializations. The figure verifies that σ^2_{NTKGP} is highly correlated with the observed output variance of the NN and the variances are generally confined within some region, which provides an empirical corroboration for our Theorem 3.1. This justifies our choice of using σ^2_{NTKGP} to measure the output variance w.r.t. the model initializations and hence the initialization robustness. In addition, in App. H.2, we present further experimental results to show that the output variance predicted using sparse GP approximations, which is more computationally efficient (Sec. 3.1.1), is also highly correlated with the observed output variance.

5.2. Experiments on Regression Tasks

Here we evaluate our EV-GP criterion[3] (Sec. 3.3) using regression tasks. In the experiments here, each algorithm is

[3]When reporting the value α_{EV}, we will ignore the $\sigma^2_{\text{NN}}(x|\emptyset)$ terms and instead report the average of $-\sigma^2_{\text{NN}}(x|\mathcal{X})$.

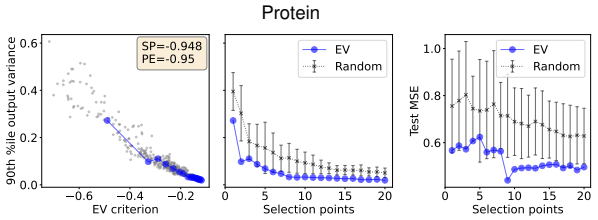

Figure 3. Results of sequential data selection in regression tasks (discussed in Sec. 5.2 and detailed descriptions in App. H.1).

given an unlabelled pool of data and no initial labelled data, and all methods use a 2-layer MLP with ReLU activation.

In Fig. 3, our EV-GP criterion is used to sequentially select the data points (i.e. the batch size is 1). In the first column of Fig. 3, we plot the 90th percentile output variance (i.e. 90% of the test points have lower output variance than this value) as the vertical axis, against the values of our EV-GP criterion as the horizontal axis. The gray dots show that our EV-GP criterion is highly correlated with output variance, and the blue dots, which display the points selected by our EV-GP criterion during active learning, demonstrate that our EV-GP criterion is able to select points which lead to low output variance (since the selected points are mostly clustered in the bottom right corner). This is also corroborated by the middle column of Fig. 3, which shows that sequentially maximizing our EV-GP criterion indeed leads to the selection points which progressively reduce the output variance, and our EV-GP criterion consistently outperform random search. The third column of Fig. 3 shows that the points selected by maximizing our EV-GP criterion also sequentially reduce the test MSE and hence improve the predictive performance of the NN. Therefore, the second and third columns of Fig. 3 combine to provide an empirical justification for our Theorem 3.3, which has theoretically shown that minimizing the approximate output variance σ_{NTKGP} (which is achieved by maximizing our EV-GP criterion) also improves the generalization performance of overparameterized NNs.

We have also tested our EV-GP criterion in the more practical active learning setting where a batch of points are selected in every round (with a batch size of 20). Fig. 4 shows that in the batch setting, our EV-GP criterion is still able select batches of points which lead to both low output variance (first column) and small test error (second column), and outperforms the other baselines. We include more experimental results for the regression tasks in App. H.3, which are consistent with those shown here (Fig. 3 and Fig. 4). A particularly interesting additional result is Fig. 12, which shows that an easier regression task leads to a larger degree of correlation between the output variance and the test error. This, interestingly, is consistent with Theorem 3.3, because it has theoretically shown that an easier task (i.e., a simpler groundtruth function which is indicated by a smaller B) reduces the value of ζ on the right hand side of Theorem 3.3, which consequently increases the degree of correlation between the output variance (i.e., right hand side) and the generalization error (i.e., left hand side).

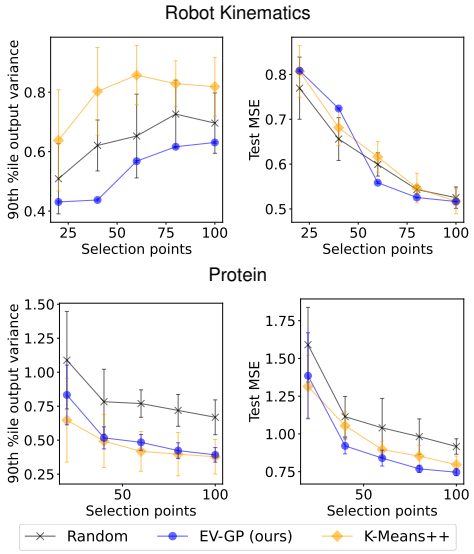

Figure 4. Results on regression tasks. *Left:* output variance of test predictions after training using the labelled active set. *Right:* test MSE. The x-axis represents the size of the selected active set. More details about the metrics are in App. G.3.

5.3. Experiments on Classification Tasks

Here we apply our EV-GP criterion to classification tasks. We use a wider variety of NN architectures, including MLPs (ReLU activation), convolutional NNs (CNNs), and WideResNets (Zagoruyko & Komodakis, 2016) which has also been used in experiments of Mohamadi et al. (2022).

Performance Comparison. Fig. 5(a-b) presents the comparison of our EV-GP criterion with other baselines, in which all methods use MLPs. The figures show that our EV-GP criterion is indeed able to select points which lead to both initialization robustness (i.e., low output entropy plotted in the first column) and good generalization performances (i.e., high test accuracy shown in the second column). The results are consistent with those for the regression tasks (Sec. 5.2). Moreover, our EV-GP criterion outperforms the other baselines in Fig. 5(a-b), especially in the earlier rounds when there is a small number of selected points. Fig. 5(c-d) plots the results using more sophisticated NN architectures (i.e., CNNs and WideResNets), in which our EV-GP criterion also consistently outperforms the other baselines in terms of the test accuracy. Further experimental results on other dataset and model architectures (such as ResNet18) are also provided in App. H.4.1.

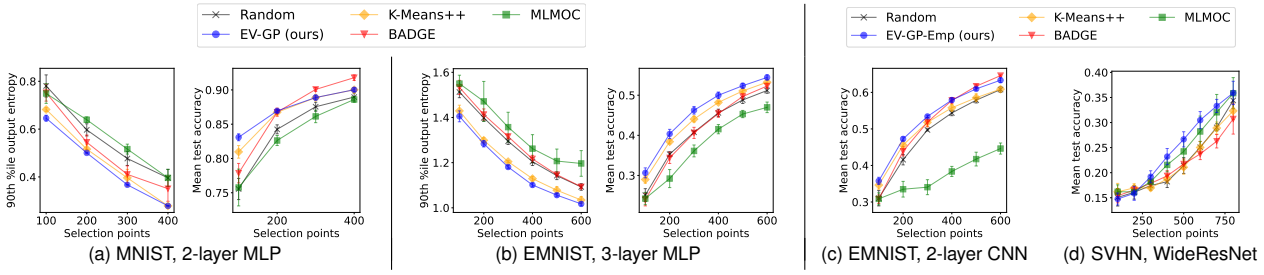

Figure 5. Results of active learning on classification tasks using NNs. (a-b) the output entropy and mean test accuracy of test predictions for experiments involving MLPs. (c-d) the mean test accuracy for experiments involving more complex models with convolutions. More details about the metrics are provided in App. G.3.

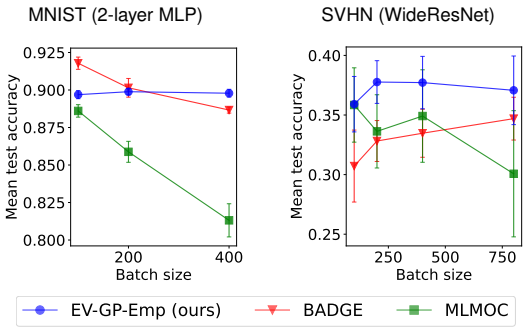

Figure 6. Results on classification with varying batch sizes.

Effects of the Batch Size. Here we examine the impact of the batch size on the performances of different active learning algorithms, by fixing the total query budget and varying the batch size. The results in Fig. 6 show that EV-GP-EMP (which uses the empirical NTK) is minimally affected by the increasing batch size[4]. This is reasonable because these algorithms do not require the labels, therefore, a smaller batch size (i.e., more frequent availability of the labels) has no impact on the performance. In contrast, when the batch size is increased, MLMOC experiences a large drop in performance for both datasets and the performance of BADGE is significantly decreased for MNIST. This may be mainly attributed to their reliance on the labels, because a larger batch size reduces the frequency of the availability of, and hence their abilities to use, labels. Moreover, another factor which causes the detrimental effect of a larger batch size on MLMOC is that a larger batch size is likely to reduce the diversity of the selected points (Mohamadi et al., 2022). We provide further discussions and additional results involving varying query batch sizes in App. H.4.3.

Computation Costs of EV-GP. We find that the cost for only running EV-GP is not significantly higher than other active learning baseline algorithms, while still being able to outperform these baselines. Nonetheless, in practical scenarios, the cost for querying labels will often dominate the cost for active data selection. Since EV-GP is training-free and minimally affected by query batch size, in practical scenarios EV-GP will be less affected by these high labelling costs and be more advantageous than other active learning algorithms. We discuss the results regarding the computation time of EV-GP further in App. H.4.4.

Effects of the Network Width. We find that using a wider NN at training time improves both the model accuracy and initialization robustness, while increasing the width of the NN for active learning (i.e., data selection) only affects the initialization robustness yet has negligible effects on the resulting model accuracy. We explore these results further in App. H.4.5.

5.4. Active Learning with Model Selection

Here we evaluate the effect of our model selection algorithm (Sec. 4), by comparing the performance of our EV-GP+MS algorithm (with model selection) with that of our EV-GP algorithm using a fixed model architecture (2-layer MLP with ReLU activation) and with model selection using NAS-WOT (Mellor et al., 2021). The left figure in Fig. 7 shows that for some datasets for which the fixed architecture already leads to a good performance, our EV-GP+MS with model selection is able to perform on par with that of EV-GP with the fixed architecture. For some other datasets where the fixed architecture is *not* adequate (e.g. when a deeper model or a different activation function can better model the data), our EV-GP+MS is able to discover a better model architecture and hence achieve a lower loss (the right figure in Fig. 7). The performance of EV-GP+MS can be attributed to the fact that it is coherent with our active learning criterion. Specifically, although it is possible to run other model selection algorithms together with EV-GP criterion, our model selection criterion (11) is developed based on the same theoretical foundation as our active learning criterion (9). Therefore, our model selection algorithm leads to a coherent framework for joint data and model selection, hence resulting in better empirical performances as shown. We provide further discussions and additional results for

[4]We omit RANDOM, K-MEANS++ and EV-GP from the graph since the selection algorithm is independent of batch size.

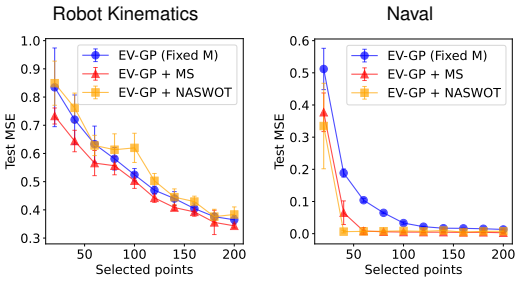

Figure 7. Results of EV-GP+MS compared to EV-GP on a fixed model architecture (2-layer MLP with ReLU activation).

EV-GP+MS in App. H.5.

6. Related Works

The existing works on active learning are usually based on either diversity or uncertainty. Diversity-based active learning algorithms aim to select a diverse subset of data, in which the diversity of the data is measured based on a discriminator (Gissin & Shalev-Shwartz, 2019; Sinha et al., 2019; Kim et al., 2021), using some latent representation (Sener & Savarese, 2018; Ash et al., 2020), or based on the degree to which they match with the other unlabelled data (Chattopadhyay et al., 2012; Shui et al., 2020). Uncertainty-based active learning algorithms select the unlabelled data based on how uncertain the trained model is about their predictions, in which the uncertainty can be captured through the softmax output (Ranganathan et al., 2017; He et al., 2019) or using a Bayesian NN (Gal et al., 2017; Kirsch et al., 2019; 2021). In addition, some previous works on active learning have also combined both diversity and uncertainty for data selection (Ash et al., 2020; Prabhu et al., 2021). Some works have improved the efficiency of neural active learning by estimating the performance of the NNs using computationally efficient proxies, such as an NN with a smaller size (Coleman et al., 2020). Another line of work is to approximate the neural network performance using the theory of NTKs (Wang et al., 2021; 2022; Mohamadi et al., 2022), which is able provide theoretical guarantees on how a model would behave without the need to perform expensive model training.

In addition to active learning, NTKs also find use in other tasks such as data valuation (Wu et al., 2022), neural architecture search (Shu et al., 2022a;c), multi-armed bandits (Dai et al., 2023), Bayesian optimization (Dai et al., 2022b), and uncertainty quantification (He et al., 2020). NTKs have also been useful in understanding model generalization loss (Cao & Gu, 2019) and also and in linking neural network training with gradient descent and kernel ridge regression (Arora et al., 2019a; Vakili et al., 2021b). Moreover, the problem of active learning is also closely related to Bayesian optimization (BO) which also utilizes the principled uncertainty measures provided by GPs to optimize a black-box function (Daxberger & Low, 2017; Kharkovskii et al., 2020b; Balakrishnan et al., 2020; Nguyen et al., 2021d;e; Dai et al., 2022a). Specifically, active learning can be viewed as a variant of BO for pure exploration.

7. Conclusion

We have introduced a computationally efficient and theoretically grounded criterion for neural active learning, which can lead to the selection of points that result in both initialization robustness and good generalization performances. Extensive empirical results have shown that our criterion is highly correlated to both the initialization robustness and generalization error, and that it consistently outperforms existing baselines. An interesting future direction is to incorporate our algorithm to select the initial points for other neural active learning algorithms to further enhance their performances, because our algorithm has shown impressive performances in scenarios with limited initial labelled data. Moreover, given the close connections between active learning and Bayesian optimization (BO) (Sec. 6), as well as the widespread real-world applications of BO, we will also explore extending our active learning algorithm to other real-world problem settings that have been considered by BO, such as problems with high-dimensional input spaces (Hoang et al., 2018), multi-fidelity observations (Zhang et al., 2017; 2019; Dai et al., 2019), delayed feedback (Verma et al., 2022), a necessity for rigorous privacy guarantees (Kharkovskii et al., 2020a) and a requirement for risk aversion (Nguyen et al., 2021a;b; Tay et al., 2022), as well as problems that fall under the federated/collaborative setting (Dai et al., 2020b; 2021; Sim et al., 2021) and game-theoretic setting (Dai et al., 2020a; Tay et al., 2023). Furthermore, it is also a promising future topic to extend our model selection method (Sec. 4) by incorporating more advanced methods for neural architecture search (Shu et al., 2022a;b).

Acknowledgements

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018) and by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Programmatic Funds (Award A20H6b0151). We would like to thank Shu Yao for his valuable inputs to our paper.

References

Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks. In *Proc. ICML*, pp. 322–332. PMLR, May 2019a.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. In *Proc. NeurIPS*, number 731, pp. 8141–8150. Curran Associates Inc., Red Hook, NY, USA, December 2019b.

Arthur, D. and Vassilvitskii, S. K-means++: The advantages of careful seeding. In *Proc. SODA*, SODA '07, pp. 1027–1035, USA, January 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-624-5.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *Proc. ICLR*, April 2020.

Balakrishnan, S., Nguyen, Q. P., Low, B. K. H., and Soh, H. Efficient exploration of reward functions in inverse reinforcement learning via Bayesian optimization. In *Proc. NeurIPS*, pp. 4187–4198, 2020.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: Composable transformations of Python+NumPy programs, 2018.

Cao, Y. and Gu, Q. Generalization Bounds of Stochastic Gradient Descent for Wide and Deep Neural Networks. In *Proc. NeurIPS*, volume 32. Curran Associates, Inc., 2019.

Chattopadhyay, R., Wang, Z., Fan, W., Davidson, I., Panchanathan, S., and Ye, J. Batch Mode Active Sampling based on Marginal Probability Distribution Matching. In *Proc. KDD*, volume 2012, pp. 741–749, 2012. doi: 10.1145/2339530.2339647.

Chowdhury, S. R. and Gopalan, A. On Kernelized Multi-armed Bandits. In *Proc. ICML*, pp. 844–853. PMLR, July 2017.

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. EMNIST: Extending MNIST to handwritten letters. In *Proc. IJCNN*, pp. 2921–2926, May 2017. doi: 10.1109/IJCNN.2017.7966217.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. Selection via Proxy: Efficient Data Selection for Deep Learning, October 2020.

Cyr, E. C., Gulian, M. A., Patel, R. G., Perego, M., and Trask, N. A. Robust Training and Initialization of Deep Neural Networks: An Adaptive Basis Viewpoint, December 2019.

Dai, Z., Yu, H., Low, B. K. H., and Jaillet, P. Bayesian optimization meets Bayesian optimal stopping. In *Proc. ICML*, pp. 1496–1506, 2019.

Dai, Z., Chen, Y., Low, B. K. H., Jaillet, P., and Ho, T.-H. R2-B2: Recursive reasoning-based Bayesian optimization for no-regret learning in games. In *Proc. ICML*, pp. 2291–2301, 2020a.

Dai, Z., Low, B. K. H., and Jaillet, P. Federated Bayesian optimization via Thompson sampling. In *Proc. NeurIPS*, pp. 9687–9699, 2020b.

Dai, Z., Low, B. K. H., and Jaillet, P. Differentially private federated Bayesian optimization with distributed exploration. In *Proc. NeurIPS*, pp. 9125–9139, 2021.

Dai, Z., Chen, Y., Yu, H., Low, B. K. H., and Jaillet, P. On provably robust meta-Bayesian optimization. In *Proc. UAI*, 2022a.

Dai, Z., Shu, Y., Low, B. K. H., and Jaillet, P. Sample-then-optimize batch neural Thompson sampling. In *Proc. NeurIPS*, 2022b.

Dai, Z., Shu, Y., Verma, A., Fan, F. X., Low, B. K. H., and Jaillet, P. Federated neural bandits. In *Proc. ICLR*, 2023.

Daxberger, E. A. and Low, B. K. H. Distributed batch Gaussian process optimization. In *Proc. ICML*, pp. 951–960, 2017.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web], 2012.

Dua, D. and Graff, C. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml.

Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., and Dean, J. A guide to deep learning in healthcare. *Nature Medicine*, 25(1):24–29, January 2019. ISSN 1546-170X. doi: 10.1038/s41591-018-0316-z.

Gal, Y., Islam, R., and Ghahramani, Z. Deep Bayesian Active Learning with Image Data. In *Proc. ICML*, pp. 1183–1192. PMLR, July 2017.

Gissin, D. and Shalev-Shwartz, S. Discriminative Active Learning. *CoRR*, abs/1907.06347, 2019.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, pp. 249–256. JMLR Workshop and Conference Proceedings, March 2010.

He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian Deep Ensembles via the Neural Tangent Kernel. In *Proc. NeurIPS*, volume 33, pp. 1010–1022. Curran Associates, Inc., 2020.

He, T., Jin, X., Ding, G., Yi, L., and Yan, C. Towards Better Uncertainty Sampling: Active Learning with Multiple Views for Deep Convolutional Neural Network. In *Proc. ICME*, pp. 1360–1365, Shanghai, China, July 2019. IEEE. ISBN 978-1-5386-9552-4. doi: 10.1109/ICME.2019.00236.

Hoang, T. N., Low, B. K. H., Jaillet, P., and Kankanhalli, M. Active learning is planning: Nonmyopic ε-Bayes-optimal active learning of Gaussian processes. In *Proc. ECML PKDD*, pp. 494–498, 2014a.

Hoang, T. N., Low, B. K. H., Jaillet, P., and Kankanhalli, M. Nonmyopic ε-Bayes-optimal active learning of Gaussian processes. In *Proc. ICML*, pp. 739–747, 2014b.

Hoang, T. N., Hoang, Q. M., and Low, B. K. H. A Unifying Framework of Anytime Sparse Gaussian Process Regression Models with Stochastic Variational Inference for Big Data. In *Proc. ICML*, pp. 569–578. PMLR, June 2015.

Hoang, T. N., Hoang, Q. M., and Low, B. K. H. Decentralized high-dimensional Bayesian optimization with factor graphs. In *Proc. AAAI*, pp. 3231–3238, 2018.

Jacot, A., Gabriel, F., and Hongler, C. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Proc. NeurIPS*, volume 31. Curran Associates, Inc., 2018.

Kassraie, P. and Krause, A. Neural Contextual Bandits without Regret. In *Proc. AISTATS*, pp. 240–278. PMLR, May 2022.

Kharkovskii, D., Dai, Z., and Low, B. K. H. Private outsourced Bayesian optimization. In *Proc. ICML*, pp. 5231–5242, 2020a.

Kharkovskii, D., Ling, C. K., and Low, B. K. H. Nonmyopic Gaussian process optimization with macro-actions. In *Proc. AISTATS*, pp. 4593–4604, 2020b.

Kim, K., Park, D., Kim, K. I., and Chun, S. Y. Task-Aware Variational Adversarial Active Learning. In *Proc. CVPR*, pp. 8162–8171, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.00807.

Kirsch, A., van Amersfoort, J., and Gal, Y. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Proc. NeurIPS*, volume 32. Curran Associates, Inc., 2019.

Kirsch, A., Rainforth, T., and Gal, Y. Test Distribution-Aware Active Learning: A Principled Approach Against Distribution Shift and Outliers, November 2021.

Koh, P. W. and Liang, P. Understanding Black-box Predictions via Influence Functions. In *Proc. ICML*, pp. 1885–1894. PMLR, July 2017.

Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI*, IJCAI'95, pp. 1137–1143, San Francisco, CA, USA, August 1995. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-363-9.

Krause, A. and Guestrin, C. Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach. In *Proc. ICML*, ICML '07, pp. 449–456, New York, NY, USA, June 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273553.

Krause, A., Singh, A., and Guestrin, C. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *The Journal of Machine Learning Research*, 9:235–284, June 2008. ISSN 1532-4435.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. 2009.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124002, December 2020. ISSN 1742-5468. doi: 10.1088/1742-5468/abc62b.

Ling, C. K., Low, K. H., and Jaillet, P. Gaussian process planning with lipschitz continuous reward functions: Towards unifying bayesian optimization, active learning, and beyond. In *Proc. AAAI*, 2016.

Liu, T. and Zenke, F. Finding trainable sparse networks through Neural Tangent Transfer. In *Proc. ICML*, pp. 6336–6347. PMLR, November 2020.

Mellor, J., Turner, J., Storkey, A., and Crowley, E. J. Neural Architecture Search without Training. In *Proc. ICML*, pp. 7588–7598. PMLR, July 2021.

Minoux, M. Accelerated greedy algorithms for maximizing submodular set functions. In Stoer, J. (ed.), *Optimization Techniques*, volume 7, pp. 234–243. Springer-Verlag, Berlin/Heidelberg, 1978. ISBN 978-3-540-08708-3. doi: 10.1007/BFb0006528.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrak, J., and Krause, A. Lazier Than Lazy Greedy. *Proceedings*

of the AAAI Conference on Artificial Intelligence, 29(1), February 2015. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v29i1.9486.

Mohamadi, M. A., Bae, W., and Sutherland, D. J. Making Look-Ahead Active Learning Strategies Feasible with Neural Tangent Kernels. In *Proc. NeurIPS*, 2022.

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., and Mitliagkas, I. A Modern Take on the Bias-Variance Tradeoff in Neural Networks. *CoRR*, abs/1810.08591, 2018.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, December 1978. ISSN 0025-5610, 1436-4646. doi: 10.1007/BF01588971.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Nguyen, Q. P., Dai, Z., Low, B. K. H., and Jaillet, P. Optimizing conditional value-at-risk of black-box functions. In *Proc. NeurIPS*, pp. 4170–4180, 2021a.

Nguyen, Q. P., Dai, Z., Low, B. K. H., and Jaillet, P. Value-at-risk optimization with Gaussian processes. In *Proc. ICML*, pp. 8063–8072, 2021b.

Nguyen, Q. P., Low, B. K. H., and Jaillet, P. An information-theoretic framework for unifying active learning problems. In *Proc. AAAI*, pp. 9126–9134, 2021c.

Nguyen, Q. P., Tay, S., Low, B. K. H., and Jaillet, P. Top-k ranking Bayesian optimization. In *Proc. AAAI*, pp. 9135–9143, 2021d.

Nguyen, Q. P., Wu, Z., Low, B. K. H., and Jaillet, P. Trusted-maximizers entropy search for efficient Bayesian optimization. In *Proc. UAI*, pp. 1486–1495, 2021e.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. Neural Tangents: Fast and Easy Infinite Neural Networks in Python, December 2019.

Prabhu, V., Chandrasekaran, A., Saenko, K., and Hoffman, J. Active Domain Adaptation via Clustering Uncertainty-weighted Embeddings. In *Proc. ICCV*, pp. 8485–8494, Montreal, QC, Canada, October 2021. IEEE. ISBN 978-1-66542-812-5. doi: 10.1109/ICCV48922.2021.00839.

Quiñonero-Candela, J. and Rasmussen, C. E. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005. ISSN 1533-7928.

Ranganathan, H., Venkateswara, H., Chakraborty, S., and Panchanathan, S. Deep active learning for image classification. In *Proc. ICIP*, pp. 3934–3938, Beijing, September 2017. IEEE. ISBN 978-1-5090-2175-8. doi: 10.1109/ICIP.2017.8297020.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. A Survey of Deep Active Learning. *ACM Computing Surveys*, 54(9):180:1–180:40, October 2021. ISSN 0360-0300. doi: 10.1145/3472291.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. In *Proc. ICLR*. OpenReview.net, 2018.

Shu, Y., Cai, S., Dai, Z., Ooi, B. C., and Low, B. K. H. NASI: Label- and Data-agnostic Neural Architecture Search at Initialization. In *Proc. ICLR*. OpenReview.net, 2022a.

Shu, Y., Chen, Y., Dai, Z., and Low, B. K. H. Neural ensemble search via Bayesian sampling. In Cussens, J. and Zhang, K. (eds.), *Proc. UAI*, volume 180 of *Proceedings of Machine Learning Research*, pp. 1803–1812. PMLR, 2022b.

Shu, Y., Dai, Z., Wu, Z., and Low, B. K. H. Unifying and Boosting Gradient-Based Training-Free Neural Architecture Search. In *Proc. NeurIPS*, 2022c.

Shui, C., Zhou, F., Gagné, C., and Wang, B. Deep Active Learning: Unified and Principled Method for Query and Training. In *Proc. AISTATS*, pp. 1308–1318. PMLR, June 2020.

Sim, R. H. L., Zhang, Y., Low, B. K. H., and Jaillet, P. Collaborative Bayesian optimization with fair regret. In *Proc. ICML*, pp. 9691–9701, 2021.

Sinha, S., Ebrahimi, S., and Darrell, T. Variational Adversarial Active Learning, October 2019.

Tay, S. S., Foo, C. S., Urano, D., Leong, R. C. X., and Low, B. K. H. Efficient distributionally robust Bayesian optimization with worst-case sensitivity. In *Proc. ICML*, 2022.

Tay, S. S., Nguyen, Q. P., Foo, C. S., and Low, B. K. H. No-regret sample-efficient Bayesian optimization for finding nash equilibria with unknown utilities. In *Proc. AISTATS*, pp. 3591–3619. PMLR, 2023.

Vakili, S., Bouziani, N., Jalali, S., Bernacchia, A., and Shiu, D.-s. Optimal Order Simple Regret for Gaussian Process Bandits. *arXiv:2108.09262 [cs, stat]*, August 2021a.

Vakili, S., Bromberg, M., Shiu, D.-S., and Bernacchia, A. Uniform Generalization Bounds for Overparameterized Neural Networks. *CoRR*, abs/2109.06099, 2021b.

van der Maaten, L. and Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605, 2008. ISSN 1533-7928.

Verma, A., Dai, Z., and Low, B. K. H. Bayesian optimization under stochastic delayed feedback. In *Proc. ICML*, pp. 22145–22167, 2022.

Wang, H., Huang, W., Wu, Z., Tong, H., Margenot, A. J., and He, J. Deep Active Learning by Leveraging Training Dynamics. *Advances in Neural Information Processing Systems*, 35:25171–25184, December 2022.

Wang, Z., Awasthi, P., Dann, C., Sekhari, A., and Gentile, C. Neural Active Learning with Performance Guarantees. In *Proc. NeurIPS*, volume 34, pp. 7510–7521. Curran Associates, Inc., 2021.

Wu, Z., Shu, Y., and Low, B. K. H. DAVINZ: Data Valuation using Deep Neural Networks at Initialization. In *Proc. ICML*, pp. 24150–24176. PMLR, June 2022.

Xu, X., Wu, Z., Verma, A., Foo, C. S., and Low, B. K. H. FAIR: Fair collaborative active learning with individual rationality for scientific discovery. In *Proc. AISTATS*, 2023.

Zagoruyko, S. and Komodakis, N. Wide Residual Networks. In Wilson, R. C., Hancock, E. R., and Smith, W. A. P. (eds.), *Proc. BMVC*. BMVA Press, 2016.

Zhang, Y., Hoang, T. N., Low, K. H., and Kankanhalli, M. Near-optimal active learning of multi-output gaussian processes. In *Proc. AAAI*, 2016.

Zhang, Y., Hoang, T. N., Low, B. K. H., and Kankanhalli, M. Information-based multi-fidelity Bayesian optimization. In *Proc. NIPS Workshop on Bayesian Optimization*, 2017.

Zhang, Y., Dai, Z., and Low, B. K. H. Bayesian optimization with binary auxiliary information. In *Proc. UAI*, pp. 1222–1232, 2019.

A. Proof of Theorem 3.1

In this section, we will show that the output variance of neural networks with ReLU activation can be represented using σ_{NTKGP}. For notation, we will let $\|\cdot\|_p$ represent the L_p-norm of a vector or matrix. When p is unspecified, we assume that we are referring to the L_2-norm.

A.1. Dual Activations

In this subsection, we first define the concept of dual activation functions (Lee et al., 2020).

Definition A.1. Let $\phi : \mathbb{R} \to \mathbb{R}$ be an activation function. Then, the dual activation function of ϕ is given by $\check{\phi} : \mathbb{R}^{2\times 2} \to \mathbb{R}$ where

$$\check{\phi}(\Lambda) = \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Lambda)}[\phi(u)\phi(v)].$$

Similarly, if we let $\phi'(x) = \dfrac{d\phi(x)}{dx}$ be the derivative of the activation function, then its dual activation $\check{\phi}'$ is defined as

$$\check{\phi}'(\Lambda) = \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Lambda)}[\phi'(u)\phi'(v)].$$

A.2. Neural Network Assumption

We make an assumption about the parametrisation of our neural network. We use a common parametrisation from other NTK-related works, which can be found in e.g., Lee et al. (2020).

Assumption A.2. Assume $f(\cdot;\theta)$ is a multilayer perceptron with L hidden layers each with width of $k_1, k_2, \ldots, k_L$. For simplicity we assume that all hidden layers have the same width. Let the input dimension of the neural network is k_0 and the output dimension is k_{L+1}. Let the neural network be parametrised as

$$\begin{aligned} h^{(0)}(x) &= x \\ h^{(\ell)}(x) &= \frac{1}{\sqrt{k_\ell}}\phi\big(W^{(\ell)}h^{(\ell-1)}(x) + b^{(\ell)}\big) && \text{for } \ell \in [1,\ldots,L] \\ f(x) &= W^{(L+1)}h^{(L)}(x) + b^{(L+1)} \end{aligned}$$

where $W_{ij}^{(\ell)} \sim \mathcal{N}(0,\sigma_W^2)$ and $b_{ij}^{(\ell)} \sim \mathcal{N}(0,\sigma_b^2)$ are model weights and biases initialized randomly from a Gaussian distribution with variances σ_W^2 and σ_b^2 respectively, and ϕ is an activation function which is scaled so that $\check{\phi}\left(\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}\right) = 1$.

There are two main changes from the standard neural network assumption.

1. The difference in the parametrisation of the hidden layer, where a multiplicative factor of $\dfrac{1}{\sqrt{k_\ell}}$ has been added. This simplifies the computation of expectation values.
2. The scaling of the activation function. In our case, we scale the activation function ϕ such that its dual activation output is fixed under a specific input. This is an additional assumption not applied in Lee et al. (2020), and is done here to simplify the computation when we have to repeatedly apply the activation function.

A.3. Relationship between $\mathcal{K}$ and Θ

We first present the relationship between $\mathcal{K}$ and Θ. The following lemma is also presented and proven in Jacot et al. (2018).

Lemma A.3. *For a neural network in the infinite-width regime, $\mathcal{K}$ and Θ can be defined recursively as*

$$\mathcal{K}^{(0)}(x,x') = x^\top x' + \sigma_b^2 \tag{12}$$

$$\Theta^{(0)}(x,x') = 0 \tag{13}$$

$$\mathcal{K}^{(\ell)}(x,x') = \check{\phi}\left(\begin{bmatrix}\mathcal{K}^{(\ell-1)}(x,x) & \mathcal{K}^{(\ell-1)}(x,x')\\ \mathcal{K}^{(\ell-1)}(x',x) & \mathcal{K}^{(\ell-1)}(x',x')\end{bmatrix}\right) + \sigma_b^2 \qquad \ell \in [1,\cdots,L+1] \tag{14}$$

$$\Theta^{(\ell)}(x,x') = \mathcal{K}^{(\ell)}(x,x') + \Theta^{(\ell-1)}(x,x')\cdot\dot{\mathcal{K}}^{(\ell-1)}(x,x') \qquad \ell \in [1,\cdots,L+1] \tag{15}$$

where we define

$$\dot{\mathcal{K}}^{(\ell)}(x,x') = \check{\phi}'\left(\begin{bmatrix}\mathcal{K}^{(\ell)}(x,x) & \mathcal{K}^{(\ell)}(x,x') \\ \mathcal{K}^{(\ell)}(x',x) & \mathcal{K}^{(\ell)}(x',x')\end{bmatrix}\right) \tag{16}$$

Notice that (15) gives a recursive formula for computing Θ. If we "unroll" this formula, we will obtain

$$\Theta(x,x') = \sum_{\ell=1}^{L+1} \mathcal{K}^{(\ell)}(x,x') \prod_{\ell'=\ell+1}^{L+1} \dot{\mathcal{K}}^{(\ell')}(x,x'). \tag{17}$$

Given (17), it is now simple to bound $\Theta(x,x')$, since all that is required is to provide bounds for each $\mathcal{K}^{(\ell)}$ and $\dot{\mathcal{K}}^{(\ell)}$ individually. This can be done by inspecting the dual activation functions.

A.4. Properties of ReLU Dual Activation Functions

For the remainder of this section, we will consider the case where the activation function is the scaled ReLU function, defined as $\phi(x) = \sqrt{2} \cdot \max(x, 0)$. Based on Lee et al. (2020), it can be shown that for ReLU activations, if $\Lambda = \begin{bmatrix} x^\top x & x^\top y \\ y^\top x & y^\top y \end{bmatrix}$, then

$$\check{\phi}(\Lambda) = \frac{1}{\pi}\|x\|\|y\|\big(\sin\theta + (\pi-\theta)\cos\theta\big)$$

and

$$\check{\phi}'(\Lambda) = \frac{\pi-\theta}{\pi}$$

where $\theta = \arccos\left(\frac{x^\top y}{\|x\|\|y\|}\right)$. Notice that the ReLU activation in this case has a $\sqrt{2}$ multiplicative factor, which scales the activation as required from above.

For convenience, define the function $\rho : [-1, 1] \to \mathbb{R}$ where

$$\rho(r) = \check{\phi}\left(\begin{bmatrix}1 & r \\ r & 1\end{bmatrix}\right) = \frac{1}{\pi}\big(\sqrt{1-r^2} + (\pi - \arccos r)\cdot r\big). \tag{18}$$

This can be thought of as the re-parametrisation of the dual activation function ϕ in the case where $\|x\| = \|y\| = 1$, and we are only specifying the cosine distance $r = \cos\theta$ between x and y. We can also define ρ' to be a similar function but on the dual activation $\check{\phi}'$ instead,

$$\rho'(r) = \check{\phi}'\left(\begin{bmatrix}1 & r \\ r & 1\end{bmatrix}\right) = \frac{\pi - \arccos r}{\pi}. \tag{19}$$

It is simple to verify that for ReLU activations,

$$\check{\phi}\left(\begin{bmatrix}\|x\|^2 & \|x\|\|y\|\cdot r \\ \|x\|\|y\|\cdot r & \|y\|^2\end{bmatrix}\right) = \|x\|\|y\|\cdot\rho(r) \tag{20}$$

and

$$\check{\phi}'\left(\begin{bmatrix}\|x\|^2 & \|x\|\|y\|\cdot r \\ \|x\|\|y\|\cdot r & \|y\|^2\end{bmatrix}\right) = \rho'(r). \tag{21}$$

For any function f, we define a notation $f^m(x) = \underbrace{(f \circ f \circ \cdots \circ f)}_{m \text{ times}}(x)$. This is thought as repeatedly applying function f to the input value m times. For our dual activation function, we can show that repeating the function input will still result in a non-decreasing function.

Lemma A.4. *For any $n \in \mathbb{N}$, $\rho^n(r)$ is non-decreasing with respect to r.*

Proof. We can prove this by induction. For the case that $n = 1$, it is simple to see that

$$\frac{d\rho}{dr} = \frac{\pi - \arccos r}{\pi}. \tag{22}$$

We can see that $\arccos r \leq \pi$, and so $\dfrac{d\rho}{dr} \geq 0$. This means that ρ is non-decreasing.

Furthermore, we are able to see that $\rho(-1) = 0$ and $\rho(1) = 1$. Since the function is non-decreasing, this means for any $r \in [-1, 1]$, it is the case that $\rho(r) \in [0, 1] \subset [-1, 1]$.

Now, assume that ρ^n is non-decreasing. Let $r_1, r_2 \in [-1, 1]$. If $r_1 \leq r_2$, then it is the case that $\rho^n(r_1) \leq \rho^n(r_2)$. Since we know $\rho^n(r_1), \rho^n(r_2) \in [-1, 1]$, we can therefore conclude that $\rho^{n+1}(r_1) \leq \rho^{n+1}(r_2)$. □

Similarly, we can show a similar claim for the dual activation with respect to ϕ'. Note that for ϕ', we do not need to show that repeated application of the function keeps the output non-decreasing.

Lemma A.5. *ρ' is non-decreasing with respect to r.*

Proof. It is simple to see that

$$\frac{d\rho'}{dr} = \frac{1}{\pi\sqrt{1 - r^2}}. \tag{23}$$

Since square roots are always positive, it follows that $d\rho'/dr \geq 0$. This means that ρ' is an increasing function. □

A.5. Bounding $\mathcal{K}$ in terms of ρ

We will first show the relationship between $\mathcal{K}$ and ρ.

Lemma A.6. *For $\ell \in [1, \ldots, L + 1]$,*

$$\mathcal{K}^{(\ell)}(x, x') = \sqrt{u^{(\ell-1)}} \cdot \rho\big(r_u^{(\ell)}\big) + \sigma_b^2 \tag{24}$$

where $u^{(\ell)} = \big(\|x\|^2 + \ell\sigma_b^2\big)\big(\|x'\|^2 + \ell\sigma_b^2\big)$ and $r_u^{(\ell)} = \dfrac{\mathcal{K}^{(\ell-1)}(x, x')}{\sqrt{u^{(\ell-1)}}}$.

Proof. We can prove this by induction. In the case that $\ell = 1$, we see that

$$\begin{aligned}
\mathcal{K}^{(1)}(x, x') &= \check{\phi}\left(\begin{bmatrix} \mathcal{K}^{(0)}(x, x) & \mathcal{K}^{(0)}(x, x') \\ \mathcal{K}^{(0)}(x', x) & \mathcal{K}^{(0)}(x', x') \end{bmatrix}\right) + \sigma_b^2 && (25)\\
&= \check{\phi}\left(\begin{bmatrix} \|x\|^2 & x^\top x' \\ x^\top x' & \|x'\|^2 \end{bmatrix}\right) + \sigma_b^2 && (26)\\
&= \|x\|\|x'\| \cdot \rho\left(\frac{x^\top x'}{\|x\|\|x'\|}\right) + \sigma_b^2 && (27)
\end{aligned}$$

For the inductive step, assume that $\mathcal{K}^{(\ell)}(x, x') = \sqrt{u^{(\ell-1)}} \cdot \rho\big(r_u^{(\ell)}\big) + \sigma_b^2$ holds. Then, we see that

$$\begin{aligned}
\mathcal{K}^{(\ell+1)}(x, x') &= \check{\phi}\left(\begin{bmatrix} \mathcal{K}^{(\ell)}(x, x) & \mathcal{K}^{(\ell)}(x, x') \\ \mathcal{K}^{(\ell)}(x', x) & \mathcal{K}^{(\ell)}(x', x') \end{bmatrix}\right) + \sigma_b^2 && (28)\\
&= \check{\phi}\left(\begin{bmatrix} \big(\|x\|^2 + (\ell-1)\sigma_b^2\big)\rho(1) + \sigma_b^2 & \mathcal{K}^{(\ell)}(x, x') \\ \mathcal{K}^{(\ell)}(x', x) & \big(\|x'\|^2 + (\ell-1)\sigma_b^2\big)\rho(1) + \sigma_b^2 \end{bmatrix}\right) + \sigma_b^2 && (29)\\
&= \check{\phi}\left(\begin{bmatrix} \|x\|^2 + \ell\sigma_b^2 & \mathcal{K}^{(\ell)}(x, x') \\ \mathcal{K}^{(\ell)}(x', x) & \|x'\|^2 + \ell\sigma_b^2 \end{bmatrix}\right) + \sigma_b^2 && (30)\\
&= \sqrt{u^{(\ell)}} \cdot \rho\left(\frac{\mathcal{K}^{(\ell)}(x, x')}{\sqrt{u^{(\ell)}}}\right) + \sigma_b^2 && (31)
\end{aligned}$$

where we use the fact that $\mathcal{K}^{(\ell)}(x, x') \leq \sqrt{\mathcal{K}^{(\ell)}(x, x) \cdot \mathcal{K}^{(\ell)}(x', x')}$. This proves the inductive step. □

The next proofs will attempt to bound the values of $\mathcal{K}^{(\ell)}(x, x')$ based on ρ.

Lemma A.7.

$$\tilde{\rho}_{-}^{(\ell)} \leq \mathcal{K}^{(\ell)}(x, x') \leq \tilde{\rho}_{+}^{(\ell)} \tag{32}$$

where

$$\tilde{\rho}_{\pm}^{(\ell)} = \begin{cases} \pm\|x\|\|x'\| & \textit{if } \ell = 0, \\ \sqrt{u^{(\ell-1)}} \cdot \rho\left(\frac{\tilde{\rho}_{\pm}^{(\ell-1)}}{\sqrt{u^{(\ell-1)}}}\right) + \sigma_b^2 & \textit{if } \ell \geq 1. \end{cases} \tag{33}$$

Proof. We can prove so by inspecting result from Lemma A.6, and using proof by induction. In the case that $\ell = 1$,

$$\mathcal{K}^{(1)}(x, x') = \|x\|\|x'\| \cdot \rho\left(\frac{x^\top x'}{\|x\|\|x'\|}\right) + \sigma_b^2 \tag{34}$$

then it is simple to show our claim in this case from the fact that $-\|x\|\|x'\| \leq x^\top x' \leq \|x\|\|x'\|$ and that ρ is monotone.

For the inductive step, assume that the claim is true for ℓ. We see that

$$\mathcal{K}^{(\ell+1)}(x, x') = \sqrt{u^{(\ell)}} \cdot \rho\left(\frac{\mathcal{K}^{(\ell)}(x, x')}{\sqrt{u^{(\ell)}}}\right) + \sigma_b^2 \tag{35}$$

$$\leq \sqrt{u^{(\ell)}} \cdot \rho\left(\frac{\tilde{\rho}_{+}^{(\ell)}}{\sqrt{u^{(\ell)}}}\right) + \sigma_b^2 \tag{36}$$

which uses the fact that ρ is non-decreasing. A similar logic can be used to show the lower bound. This proves the inductive step and hence proves our lemma. □

Corollary A.8. *For all $\ell \in [1, \ldots, L+1]$,*

$$\sqrt{u^{(\ell-1)}} \cdot \rho\big(\hat{r}_{-}^{(\ell)}\big) + \sigma_b^2 \leq \mathcal{K}^{(\ell)}(x, x') \leq \sqrt{u^{(\ell-1)}} \cdot \rho\big(\hat{r}_{+}^{(\ell)}\big) + \sigma_b^2 \tag{37}$$

where

$$\hat{r}_{+}^{(\ell)} = 1 \tag{38}$$

and

$$\hat{r}_{-}^{(\ell)} = \begin{cases} -1 & \textit{if } \ell = 1, \\ \rho\big(\hat{r}_{-}^{(\ell-1)}\big) \cdot \frac{\ell-2}{\ell-1} & \textit{if } \ell > 1 \end{cases} \tag{39}$$

Proof. Notice that by Lemma A.7 and due to the non-decreasing nature of ρ, proving the corollary above is equivalent to showing that $\frac{\tilde{\rho}_{+}^{(\ell-1)}}{\sqrt{u^{(\ell-1)}}} \leq r_{+}^{(\ell)}$ and $\frac{\tilde{\rho}_{-}^{(\ell-1)}}{\sqrt{u^{(\ell-1)}}} \geq r_{-}^{(\ell)}$.

We will start by showing that $\frac{\tilde{\rho}_{+}^{(\ell-1)}}{\sqrt{u^{(\ell-1)}}} \leq r_{+}^{(\ell)} = 1$. It is simple to show that the statement is true for $\ell = 1$. In the case that $\ell > 1$, since $\rho(r) \leq 1$ for all r, we can show that

$$\frac{\big(\tilde{\rho}_{+}^{(\ell-1)}\big)^2}{u^{(\ell-1)}} \leq \frac{\big(\sqrt{u^{(\ell-2)}} + \sigma_b^2\big)^2}{u^{(\ell-1)}} \tag{40}$$

$$= \frac{\big(\|x\| + (\ell-2)\sigma_b^2\big)\big(\|x'\| + (\ell-2)\sigma_b^2\big) + \sigma_b^2\sqrt{\big(\|x\| + (\ell-2)\sigma_b^2\big)\big(\|x'\| + (\ell-2)\sigma_b^2\big)} + \sigma_b^4}{\big(\|x'\| + (\ell-2)\sigma_b^2\big) + \sigma_b^2\big[\big(\|x\| + (\ell-2)\sigma_b^2\big) + \big(\|x'\| + (\ell-2)\sigma_b^2\big)\big] + \sigma_b^4} \tag{41}$$

$$\leq 1 \tag{42}$$

from the inequality $2\sqrt{ab} \leq a + b$. This therefore means $\frac{\hat{\rho}_{+}^{(\ell-1)}}{\sqrt{u^{(\ell-1)}}} \leq 1$. This proves the first part of the corollary.

For the second part, we will prove so by induction. It is easy to show that the statement is true for $\ell = 1$. For the inductive step, consider some value of $\ell > 2$, and assume that the statement holds for $\ell - 1$, meaning that $\frac{\tilde{\rho}_-^{(\ell-2)}}{\sqrt{u^{(\ell-2)}}} \geq r_-^{(\ell-1)}$. Then, we can see that

$$\frac{\tilde{\rho}_-^{(\ell-1)}}{\sqrt{u^{(\ell-1)}}} = \frac{\sqrt{u^{(\ell-2)}} \cdot \rho\left(\frac{\tilde{\rho}_-^{(\ell-2)}}{\sqrt{u^{(\ell-2)}}}\right) + \sigma_b^2}{\sqrt{u^{(\ell-1)}}} \tag{43}$$

$$\geq \frac{\sqrt{u^{(\ell-2)}} \cdot \rho\big(r_-^{(\ell-1)}\big) + \sigma_b^2}{\sqrt{u^{(\ell-1)}}} \tag{44}$$

$$\geq \rho\big(r_-^{(\ell-1)}\big) \cdot \frac{\sqrt{u^{(\ell-2)}} + \sigma_b^2}{\sqrt{u^{(\ell-1)}}} \tag{45}$$

$$\geq \rho\big(r_-^{(\ell-1)}\big) \cdot \frac{\ell-2}{\ell-1} \tag{46}$$

$$= r_-^{(\ell)} \tag{47}$$

where in (46) we use the fact that

$$\frac{\sqrt{u^{(\ell-2)}} + \sigma_b^2}{\sqrt{u^{(\ell-1)}}} \geq \frac{\sqrt{u^{(\ell-2)}}}{\sqrt{u^{(\ell-1)}}} \geq \sqrt{\frac{\|x\| + (\ell-2)\sigma_b^2}{\|x\| + (\ell-1)\sigma_b^2}} \cdot \sqrt{\frac{\|x'\| + (\ell-2)\sigma_b^2}{\|x'\| + (\ell-1)\sigma_b^2}} \geq \frac{\ell-2}{\ell-1}. \tag{48}$$

This proves the second part of our corollary. □

A.6. Bounding $\dot{\mathcal{K}}$ in terms of ρ

We will first show that we can express $\dot{\mathcal{K}}$ in terms of ρ'.

Lemma A.9. *For $\ell \in [1, \ldots, L+1]$,*

$$\dot{\mathcal{K}}^{(\ell)}(x, x') = \rho'\big(r_u^{(\ell)}\big) \tag{49}$$

where $r_u^{(\ell)}$ is defined in Lemma A.6.

Proof. We can show that

$$\dot{\mathcal{K}}^{(\ell)}(x, x') = \check{\phi}'\left(\begin{bmatrix} \mathcal{K}^{(\ell-1)}(x,x) & \mathcal{K}^{(\ell-1)}(x,x') \\ \mathcal{K}^{(\ell-1)}(x',x) & \mathcal{K}^{(\ell-1)}(x',x') \end{bmatrix}\right) \tag{50}$$

$$= \check{\phi}'\left(\begin{bmatrix} \big(\|x\|^2 + (\ell-2)\sigma_b^2\big)\rho(1) + \sigma_b^2 & \mathcal{K}^{(\ell-1)}(x,x') \\ \mathcal{K}^{(\ell-1)}(x',x) & \big(\|x'\|^2 + (\ell-2)\sigma_b^2\big)\rho(1) + \sigma_b^2 \end{bmatrix}\right) \tag{51}$$

$$= \check{\phi}'\left(\begin{bmatrix} \|x\|^2 + (\ell-1)\sigma_b^2 & \mathcal{K}^{(\ell-1)}(x,x') \\ \mathcal{K}^{(\ell-1)}(x',x) & \|x'\|^2 + (\ell-1)\sigma_b^2 \end{bmatrix}\right) \tag{52}$$

$$= \rho'\left(\frac{\mathcal{K}^{(\ell-1)}(x,x')}{\sqrt{u^{(\ell-1)}}}\right). \tag{53}$$

□

Using the result above, we can now bound the values of $\dot{\mathcal{K}}^{(\ell)}(x, x')$.

Corollary A.10.

$$\rho'\big(\hat{r}_-^{(\ell)}\big) \leq \dot{\mathcal{K}}^{(\ell)}(x, x') \leq \rho'\big(\hat{r}_+^{(\ell)}\big) \tag{54}$$

where $\hat{r}_-^{(\ell)}$ and $\hat{r}_+^{(\ell)}$ are as defined in Corollary A.8.

Proof. From Corollary A.8, it is the case that $\hat{r}_-^{(\ell)} \leq r_u^{(\ell)} \leq \hat{r}_+^{(\ell)}$. The corollary then follows from this fact. □

A.7. Ratio between $\mathcal{K}$ and Θ

We will now prove the bound of the ratio between $\mathcal{K}$ and Θ.

Theorem A.11. *For a neural network with $L \geq 2$, if $\max\left\{\|x\|, \|x'\|\right\} \leq B$, then*

$$1 + \sum_{\ell=1}^{L} \rho\big(\hat{r}_-^{(\ell)}\big) \cdot \frac{\ell - 1}{L+1} \prod_{\ell'=\ell+1}^{L+1} \rho'\big(\hat{r}_-^{(\ell')}\big) \leq \frac{\Theta(x,x')}{\mathcal{K}(x,x')} \leq 1 + \frac{L}{\rho\big(\hat{r}_-^{(L+1)}\big)} \tag{55}$$

Proof. We will first prove the right hand inequality. We see that

$$\frac{\Theta(x,x')}{\mathcal{K}(x,x')} = \frac{1}{\mathcal{K}^{(L+1)}(x,x')} \cdot \sum_{\ell=1}^{L+1} \mathcal{K}^{(\ell)}(x,x') \prod_{\ell'=\ell+1}^{L+1} \dot{\mathcal{K}}^{(\ell')}(x,x') \tag{56}$$

$$\leq 1 + \sum_{\ell=1}^{L} \frac{\sqrt{u^{(\ell-1)}} \cdot \rho\big(\hat{r}_+^{(\ell)}\big) + \sigma_b^2}{\sqrt{u^{(L)}} \cdot \rho\big(\hat{r}_-^{(L+1)}\big) + \sigma_b^2} \prod_{\ell' \leq \ell+1}^{L+1} \rho'\big(\hat{r}_+^{(\ell')}\big) \tag{57}$$

$$= 1 + \sum_{\ell=1}^{L} \frac{1}{\rho\big(\hat{r}_-^{(L+1)}\big)} \cdot \frac{\sqrt{u^{(\ell-1)}} + \sigma_b^2}{\sqrt{u^{(L)}}} \tag{58}$$

$$\leq 1 + \frac{L}{\rho\big(\hat{r}_-^{(L+1)}\big)} \tag{59}$$

where we use the fact that $\frac{\sqrt{u^{(\ell-1)}} + \sigma_b^2}{\sqrt{u^{(L)}}} \leq 1$ based on a similar argument used in (41).

Similarly for the left hand inequality,

$$\frac{\Theta(x,x')}{\mathcal{K}(x,x')} \geq 1 + \sum_{\ell=1}^{L} \frac{\sqrt{u^{(\ell-1)}} \cdot \rho\big(\hat{r}_-^{(\ell)}\big) + \sigma_b^2}{\sqrt{u^{(L)}} \cdot \rho\big(\hat{r}_+^{(L+1)}\big) + \sigma_b^2} \prod_{\ell'=\ell+1}^{L+1} \rho'\big(\hat{r}_-^{(\ell')}\big) \tag{60}$$

$$\geq 1 + \sum_{\ell=1}^{L} \rho\big(\hat{r}_-^{(\ell)}\big) \cdot \frac{\sqrt{u^{(\ell-1)}}}{\sqrt{u^{(L)}} + \sigma_b^2} \prod_{\ell'=\ell+1}^{L+1} \rho'\big(\hat{r}_-^{(\ell')}\big) \tag{61}$$

$$\geq 1 + \sum_{\ell=1}^{L} \rho\big(\hat{r}_-^{(\ell)}\big) \cdot \frac{\ell - 1}{L+1} \prod_{\ell'=\ell+1}^{L+1} \rho'\big(\hat{r}_-^{(\ell')}\big) \tag{62}$$

where we use the fact that $\frac{\sqrt{u^{(\ell-1)}}}{\sqrt{u^{(L)}} + \sigma_b^2} \geq \frac{\sqrt{u^{(\ell-1)}}}{\sqrt{u^{(L+1)}}} \geq \frac{\ell-1}{L+1}$ based on a similar reasoning as (48). This proves our theorem. □

From Theorem A.12, we are therefore able to give a bound for the ratio between Θ and $\mathcal{K}$. While the constant is defined recursively based on the function ρ, we claim that this is still useful since the constants are expressed in a form which allows it to be computed directly, and more importantly, we are able to see that such a constant will exist.

A.8. Ratio between $\mathcal{K}$ and Θ when $\sigma_b = 0$

In the case that there is no bias, we can improve the bound on the ratio between $\mathcal{K}$ and Θ. The key fact is that in the case without bias, $u^{(\ell)}$ is equal for all values of ℓ, and we can simplify $\hat{\rho}_\pm^{(\ell)} = \|x\|\|x'\| \cdot \rho^\ell(\pm 1)$. During the computation, the constants $\|x\|\|x'\|$ will then cancel each other out.

Theorem A.12. *For a neural network with $L \geq 2$ and $\sigma_b = 0$,*

$$\sum_{\ell=1}^{L+1} \frac{\rho^\ell(-1)}{\rho^{L+1}(1)} \prod_{\ell'=\ell+1}^{L+1} (\rho' \circ \rho^{\ell'})(-1) \leq \frac{\Theta(x,x')}{\mathcal{K}(x,x')} \leq \sum_{\ell=1}^{L+1} \frac{\rho^\ell(1)}{\rho^{L+1}(-1)} \prod_{\ell'=\ell+1}^{L+1} (\rho' \circ \rho^{\ell'})(1) \tag{63}$$

Proof. It is easy to see from the recursive form in (14) that if $\sigma_b = 0$, then $\mathcal{K}^{(\ell)}(x, x') = \|x\|\|x'\| \cdot \rho^\ell\left(\frac{x^\top x'}{\|x\|\|x'\|}\right)$. Since we know ρ^ℓ is non-decreasing from Lemma A.4, we are able to bound

$$\|x\|\|x'\| \cdot \rho^\ell(-1) \leq \mathcal{K}^{(\ell)}(x, x') \leq \|x\|\|x'\| \cdot \rho^\ell(1). \tag{64}$$

Similarly, we can also see from (16) that $\dot{\mathcal{K}}^{(\ell)}(x, x') = (\rho' \circ \rho^\ell)\left(\frac{x^\top x'}{\|x\|\|x'\|}\right)$. Since ρ' is non-decreasing based on Lemma A.5, we are able to bound

$$(\rho' \circ \rho^\ell)(-1) \leq \dot{\mathcal{K}}^{(\ell)}(x, x') \leq (\rho' \circ \rho^\ell)(1). \tag{65}$$

Given that we can write $\Theta(x, x')$ with the recursive form given by (17), it is then simple to use (64) and (65) to bound $\frac{\Theta(x, x')}{\mathcal{K}(x, x')}$. □

A.9. Bounding the Difference Between σ_{NN} and σ_{NTKGP}

From above, we are able to see that it is possible to bound the ratio

$$a_- \leq \frac{\mathcal{K}(x, x')}{\Theta(x, x')} \leq a_+ \tag{66}$$

for some appropriately set a_- and a_+ according to either Theorem A.11 or Theorem A.12 depending on the value of σ_b (note that in the two theorems above we show the bounds for the reciprocal of what is stated in (66)). Given this bound, we are able to show the following main result.

Theorem A.13 (Formal Version of Theorem 3.1)**.** *For a neural network with ReLU activation and $L \geq 2$ hidden layers, if $\Theta_\mathcal{X} \succeq 0$, then*

$$\left|\sigma^2_{NN}(x|\mathcal{X}) - \alpha \cdot \sigma^2_{NTKGP}(x|\mathcal{X})\right| \leq \beta \tag{67}$$

where n is the upper limit on the size of the training set, $B \geq |\Theta(x, x')|$, $\alpha \in [a_-, a_+]$, $\gamma = B \cdot \max\left\{\alpha - a_-, a_+ - \alpha\right\}$, and $\beta = \gamma + \frac{n\gamma B^2}{\lambda_{\min}(\Theta_\mathcal{X})^2} + \frac{2n\gamma B}{\lambda_{\min}(\Theta_\mathcal{X})}$.

Proof. First, we know that we have $\frac{\mathcal{K}(x, x')}{\Theta(x, x')} \in [a_-, a_+]$ by assumption. In the case that $\Theta(x, x') \geq 0$, we can convert the multiplicative bound into an additive bound as

$$\begin{aligned}
\mathcal{K}(x, x') &\geq a_- \cdot \Theta(x, x') && (68)\\
\alpha \cdot \Theta(x, x') - \mathcal{K}(x, x') &\leq (\alpha - a_-) \cdot \Theta(x, x') && (69)\\
&\leq (\alpha - a_-) \cdot B && (70)
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{K}(x, x') &\leq a_+ \Theta(x, x') && (71)\\
\mathcal{K}(x, x') - \alpha \cdot \Theta(x, x') &\leq (\alpha - a_+) \cdot \Theta(x, x') && (72)\\
&\leq (\alpha - a_+) \cdot B && (73)\\
& && (74)
\end{aligned}$$

which can be combined to give $\left|\mathcal{K}(x, x') - \alpha \cdot \Theta(x, x')\right| \leq \gamma$ for γ as defined earlier. The same is the case when $\Theta(x, x') < 0$.

Given this additive bound, we are then able to bound each term which appears in σ_{NN} individually. We can see that

$$\begin{aligned}
\left|\mathcal{K}_{x\mathcal{X}}\Theta_\mathcal{X}^{-1}\Theta_{\mathcal{X}x} - \alpha \cdot \Theta_{x\mathcal{X}}\Theta_\mathcal{X}^{-1}\Theta_{\mathcal{X}x}\right| &= \left|\left(\mathcal{K}_{x\mathcal{X}} - \alpha \cdot \Theta_{x\mathcal{X}}\right)\Theta_\mathcal{X}^{-1}\Theta_{\mathcal{X}x}\right| && (75)\\
&\leq \lambda_{\max}(\Theta_\mathcal{X}^{-1})\|\mathcal{K}_{\mathcal{X}x} - \alpha \cdot \Theta_{\mathcal{X}x}\|\|\Theta_{\mathcal{X}x}\| && (76)\\
&\leq \frac{1}{\lambda_{\min}(\Theta_\mathcal{X})} \cdot \gamma\sqrt{n} \cdot B\sqrt{n} && (77)\\
&= \frac{n\gamma B}{\lambda_{\min}(\Theta_\mathcal{X})}. && (78)
\end{aligned}$$

Similarly,

$$\begin{aligned}\left|\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} - \alpha\cdot\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\right| &= \left|\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}(\mathcal{K}_{\mathcal{X}} - \alpha\cdot\Theta_{\mathcal{X}})\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\right| && (79)\\ &\leq \lambda_{\max}(\mathcal{K}_{\mathcal{X}} - \alpha\cdot\Theta_{\mathcal{X}})\|\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\|^2 && (80)\\ &= \lambda_{\max}(\mathcal{K}_{\mathcal{X}} - \alpha\cdot\Theta_{\mathcal{X}})\cdot\lambda_{\max}(\Theta_{\mathcal{X}}^{-1})^2\cdot\|\Theta_{\mathcal{X}x}\|^2 && (81)\\ &\leq \frac{n\gamma B^2}{\lambda_{\min}(\Theta_{\mathcal{X}})^2}. && (82)\end{aligned}$$

Combining these results together, we obtain

$$\begin{aligned}\left|\sigma_{\text{NN}}^2(x|\mathcal{X}) - \alpha\cdot\sigma_{\text{NTKGP}}^2(x|\mathcal{X})\right| &= \Big|\big(\mathcal{K}_x + \Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} - \mathcal{K}_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} - \Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}x}\big)+\\ &\qquad - \alpha\cdot\big(\Theta_x - \Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\big)\Big| && (83)\\ &\leq \left|\mathcal{K}_x - \alpha\Theta_x\right| + \left|\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} - \alpha\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\right|\\ &\qquad + \left|\mathcal{K}_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} - \alpha\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\right| + \left|\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}x} - \alpha\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\right| && (84)\\ &\leq \gamma + \frac{n\gamma B^2}{\lambda_{\min}(\Theta_{\mathcal{X}})^2} + \frac{2n\gamma B}{\lambda_{\min}(\Theta_{\mathcal{X}})} && (85)\end{aligned}$$

which proves our claim. □

A.10. Bounding the Ratio Between α_{EV} using σ_{NN} and σ_{NTKGP}

It turns out that we are able to get a bound on the ratio of the EV criterion directly when we use σ_{NN} versus when we use σ_{NTKGP}. Considering the case of a single test point, we can see that using σ_{NN}, the EV criterion gives

$$\alpha_{\text{EV,NN}}(x|\mathcal{X}) = \sigma_{\text{NN}}^2(x|\emptyset) - \sigma_{\text{NN}}^2(x|\mathcal{X}) = 2\cdot\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}x} - \Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} \geq 0, \tag{86}$$

and using σ_{NTKGP},

$$\alpha_{\text{EV,NTKGP}}(x|\mathcal{X}) = \sigma_{\text{NTKGP}}^2(x|\emptyset) - \sigma_{\text{NTKGP}}^2(x|\mathcal{X}) = \Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} \geq 0. \tag{87}$$

Based on this, we are able to show the following bound.

Lemma A.14. *Assume $\mathcal{X}$ is such that $\Theta_{\mathcal{X}}, \mathcal{K}_{\mathcal{X}} \succeq 0$. Let $\|\Theta_{\mathcal{X}}\|_\infty, \|\Theta_{\mathcal{X}x}\|_\infty \leq B$ and $a_- \leq \dfrac{\mathcal{K}(x,x')}{\Theta(x,x')} \leq a_+$. Then,*

$$\frac{2a_-\cdot\lambda_{\min}(\Theta_{\mathcal{X}})}{B} - \frac{a_+B}{\lambda_{\min}(\Theta_{\mathcal{X}})} \leq \frac{\alpha_{\text{EV,NN}}(x|\mathcal{X})}{\alpha_{\text{EV,NTKGP}}(x|\mathcal{X})} \leq \frac{2a_+B}{\lambda_{\min}(\Theta_{\mathcal{X}})}. \tag{88}$$

Proof. For the right hand inequality, we can see that

$$\begin{aligned}\frac{\alpha_{\text{EV,NN}}(x|\mathcal{X})}{\alpha_{\text{EV,NTKGP}}(x|\mathcal{X})} &\leq \frac{2\cdot\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}x}}{\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}} && (89)\\ &\leq \frac{2\cdot\lambda_{\max}(\Theta_{\mathcal{X}}^{-1})\|\Theta_{\mathcal{X}x}\|\|\mathcal{K}_{\mathcal{X}x}\|}{\lambda_{\min}(\Theta_{\mathcal{X}}^{-1})\|\Theta_{\mathcal{X}x}\|^2} && (90)\\ &= \frac{2\cdot\lambda_{\max}(\Theta_{\mathcal{X}})\|\mathcal{K}_{\mathcal{X}x}\|}{\lambda_{\min}(\Theta_{\mathcal{X}})\|\Theta_{\mathcal{X}x}\|} && (91)\\ &\leq \frac{2a_+B}{\lambda_{\min}(\Theta_{\mathcal{X}})}. && (92)\end{aligned}$$

Meanwhile, for the left hand inequality,

$$\frac{\alpha_{\text{EV,NN}}(x|\mathcal{X})}{\alpha_{\text{EV,NTKGP}}(x|\mathcal{X})} = \frac{2\cdot\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}x}}{\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}} - \frac{\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}}{\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}} \tag{93}$$

$$\geq \frac{2\cdot\lambda_{\min}(\Theta_{\mathcal{X}}^{-1})\|\Theta_{\mathcal{X}x}\|\|\mathcal{K}_{\mathcal{X}x}\|}{\lambda_{\max}(\Theta_{\mathcal{X}}^{-1})\|\Theta_{\mathcal{X}x}\|^2} - \frac{\lambda_{\max}(\mathcal{K}_{\mathcal{X}})\|\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\|^2}{\lambda_{\min}(\Theta_{\mathcal{X}}^{-1})\|\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x}\|^2} \tag{94}$$

$$\geq \frac{2\cdot\lambda_{\min}(\Theta_{\mathcal{X}}^{-1})\|\mathcal{K}_{\mathcal{X}x}\|}{\lambda_{\max}(\Theta_{\mathcal{X}}^{-1})\|\Theta_{\mathcal{X}x}\|} - \frac{\lambda_{\max}(\mathcal{K}_{\mathcal{X}})}{\lambda_{\min}(\Theta_{\mathcal{X}})} \tag{95}$$

$$\geq \frac{2\cdot\lambda_{\min}(\Theta_{\mathcal{X}})\|\mathcal{K}_{\mathcal{X}x}\|}{\lambda_{\max}(\Theta_{\mathcal{X}})\|\Theta_{\mathcal{X}x}\|} - \frac{\lambda_{\max}(\mathcal{K}_{\mathcal{X}})}{\lambda_{\min}(\Theta_{\mathcal{X}})} \tag{96}$$

$$\geq \frac{2\cdot\lambda_{\min}(\Theta_{\mathcal{X}})\cdot a_-}{B} - \frac{a_+B}{\lambda_{\min}(\Theta_{\mathcal{X}})} \tag{97}$$

which provides a lower bound. □

This provides another method of showing the agreement when using σ^2_{NTKGP} variance function for our criterion compared to using σ^2_{NN} variance function.

B. Proof of Theorem 3.3

In this section we will state Theorem B.1 more formally and proceed to prove it.

Theorem B.1 (Formal version of Theorem 3.3)**.** *Let the function $f^* \in \mathcal{H}_\Theta$ and training set $(\mathcal{X}, \mathbf{y})$ follow Assumption 3.2. Suppose we train an infinitely wide neural network $f(\cdot;\theta)$ with training dataset $(\mathcal{X}, \mathbf{y})$ and mean-squared error loss function as given by* (1)*, using gradient descent until convergence. Then, for any $x \in \mathcal{X}$, with probability at least $1-2\delta$ over the random observation noise and network initialization,*

$$\left|f^*(x) - f(x;\text{train}(\theta_0))\right| \leq \left[B + \left(\frac{R}{\lambda}+1\right)\sqrt{2\log\delta^{-1}}\right]\sigma_{\text{NTKGP}}(x|\mathcal{X}) \tag{98}$$

where λ is the regularization of the loss function.

Proof. By the triangle inequality, we can show that

$$\left|f^*(x) - f(x;\text{train}(\theta_0))\right| \leq \underbrace{\left|f^*(x) - \mu(x|\mathcal{X})\right|}_{①} + \underbrace{\left|\mu(x|\mathcal{X}) - f(x;\theta_\infty)\right|}_{②} \tag{99}$$

$$\leq \left(B + \frac{R}{\lambda}\sqrt{2\log\delta^{-1}}\right)\sigma_{\text{NTKGP}}(x|\mathcal{X}) + \left|\mu(x|\mathcal{X}) - f(x;\theta_\infty)\right| \tag{100}$$

$$\leq \left(B + \frac{R}{\lambda}\sqrt{2\log\delta^{-1}}\right)\sigma_{\text{NTKGP}}(x|\mathcal{X}) + \sqrt{2\log\delta^{-1}}\cdot\sigma_{\text{NN}}(x|\mathcal{X}) \tag{101}$$

$$\leq \left[B + \left(\frac{R}{\lambda}+1\right)\sqrt{2\log\delta^{-1}}\right]\sigma_{\text{NTKGP}}(x|\mathcal{X}) \tag{102}$$

where

- (99) is true due to the triangle inequality,
- (100) is true with probability at least $1-\delta$ based on Theorem 1 of (Vakili et al., 2021a),
- (101) is true with probability at least $1-\delta$ due to Hoeffding inequality for sub-Gaussian random variables, and
- (102) is true since we know that $\sigma_{\text{NTKGP}}(x|\mathcal{X}) \geq \sigma_{\text{NN}}(x|\mathcal{X})$ from He et al. (2020).

By union bound, the statement above is true with probability at least $1-2\delta$. □

The loss as decomposed in (99) can also be thought of as the sum of the model bias (i.e. how well our neural network architecture is able to fit the given underlying function), given by ①, and the variance of prediction due to the random network initialization, given by ②. By Theorem B.1, we can show that we are minimising the predictive variance ②, however our ability to do so will also depend on the bias term ①.

C. Efficient Computations of the NTKGP

In the practical deployment of active learning, the cost of data label querying is usually significantly costly (in terms of time and computation costs), meaning the time cost of the active set selection can be easily be dominated by the querying costs. Note that our algorithm incurs less cost of data label querying because it can run in large batches while maintaining competitive performance. Furthermore, another factor contributing significantly to the cost of active learning is the training of the neural networks, which our algorithm mitigates due to its training-free nature. Nonetheless, we preent some possible speedups that our algorithm can utilize, particularly in computing the value of σ^2_{NTKGP}.

C.1. Incremental Computation of σ^2_{NTKGP}

Computing $\sigma^2_{\text{NTKGP}}(\cdot|\mathcal{X})$ for the active learning criteria is computationally expensive. Fortunately, we know that in our active learning algorithm the labelled set $\mathcal{X}$ is always incremented by one each time, and so when we would like to compute the criterion $\alpha(\mathcal{X} \cup \{x'\})$, rather than recomputing the $\sigma^2_{\text{NTKGP}}(\cdot|\mathcal{X} \cup \{x'\})$ again each time, we can just get the updated value of $\sigma^2_{\text{NTKGP}}(\cdot|\mathcal{X} \cup \{x'\})$ based on the already-known value of $\sigma^2_{\text{NTKGP}}(\cdot|\mathcal{X})$.

Suppose we let $\mathcal{X}' = \mathcal{X} \cup \{x'\}$. We can utilize the formula for inversion of block matrices (e.g. Eqns. A.11 and A.12 of of Rasmussen & Williams (2006))

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & -\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & \left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \end{bmatrix} \tag{103}$$

and the matrix inversion lemma (e.g. Eqn. A.9 of Rasmussen & Williams (2006))

$$(\mathbf{A} + \mathbf{B}\mathbf{C}\mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1} \tag{104}$$

to see that

$$\Theta_{\mathcal{X}'}^{-1} = \begin{bmatrix} \Theta_{\mathcal{X}} & \Theta_{\mathcal{X}x'} \\ \Theta_{x'\mathcal{X}} & \Theta_{x'} \end{bmatrix}^{-1} = \begin{bmatrix} \left(\Theta_{\mathcal{X}} - \Theta_{\mathcal{X}x'}\Theta_{x'}^{-1}\Theta_{x'\mathcal{X}}\right)^{-1} & -\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\left(\Theta_{x'} - \Theta_{x'\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\right)^{-1} \\ -\Theta_{x'}^{-1}\Theta_{x'\mathcal{X}}\left(\Theta_{\mathcal{X}} - \Theta_{\mathcal{X}x'}\Theta_{x'}^{-1}\Theta_{x'\mathcal{X}}\right)^{-1} & \left(\Theta_{x'} - \Theta_{x'\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\right)^{-1} \end{bmatrix} \tag{105}$$

and

$$\begin{aligned} \sigma^2_{\text{NTKGP}}(x|\mathcal{X}') &= \Theta_x - \Theta_{x\mathcal{X}'}\Theta_{\mathcal{X}'}^{-1}\Theta_{\mathcal{X}'x} \qquad (106) \\ &= \sigma^2_{\text{NTKGP}}(x|\mathcal{X}) - \Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\left(\Theta_{x'} - \Theta_{x'\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\right)^{-1}\Theta_{x'\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x} \\ &\quad + 2\cdot\Theta_{x\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\left(\Theta_{x'} - \Theta_{x'\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\right)^{-1}\Theta_{x'x} \\ &\quad - \Theta_{xx'}\left(\Theta_{x'} - \Theta_{x'\mathcal{X}}\Theta_{\mathcal{X}}^{-1}\Theta_{\mathcal{X}x'}\right)^{-1}\Theta_{x'x}. \qquad (107) \end{aligned}$$

Given that $\Theta_{\mathcal{X}}^{-1}$ can also be reused from the previous round and updated iteratively using Eq. (105), this means the variance can be computed more efficiently.

C.2. Approximating μ_{NTKGP} and Σ_{NTKGP} Using Sparse GPs

A benefit for using Σ_{NTKGP} for our criterion is that unlike Σ_{NN}, the covariance matrices in the form of Σ_{NTKGP} is well-studied in Gaussian process literature, and methods of sparsifying the kernel matrix has been proposed. Below, we discuss how such approximation techniques can be used.

Given we have a set of labelled training data $\mathcal{X}$, we would like to compute the posterior variance $\Sigma_{\text{NTKGP}}(\mathcal{X}_T|\mathcal{X})$. Since it is expensive to compute the posterior directly, we will use techniques from sparse GPs to compute the approximate posterior. In particular, we will use the approximation framework as proposed by Hoang et al. (2015).

Let $\mathcal{U} = (\mathcal{X}_\mathcal{U}, \mathbf{y}_\mathcal{U})$ be a set of inducing points which is a representative for $\mathcal{X}_T$. Note that in general, $\mathcal{U}$ does not have to be a subset of $\mathcal{X}$, and therefore is not necessarily a valid candidate for the active set. For simplicity, we will attempt to apply the Fully-Independent Conditional (FIC) approximation, where we assume that $\mathbf{y}_T$ and $\mathbf{y}$ are conditionally independent given $\mathbf{y}_\mathcal{U}$, i.e.

$$p(\mathbf{y}_T|\mathbf{y}) = \int p(\mathbf{y}_T|\mathbf{y}_\mathcal{U}, \mathbf{y})p(\mathbf{y}_\mathcal{U}|\mathbf{y})d\mathbf{y}_\mathcal{U} \approx \int p(\mathbf{y}_T|\mathbf{y}_\mathcal{U})p(\mathbf{y}_\mathcal{U}|\mathbf{y})d\mathbf{y}_\mathcal{U}. \tag{108}$$

We can assume that $p(\mathbf{y}_\mathcal{U}|\mathbf{y}) \approx q(\mathbf{y}_\mathcal{U})$ and the goal is to compute the distribution of $q(\mathbf{y}_\mathcal{U})$ such that $D_{KL}[q(\mathbf{y}_\mathcal{U})\|p(\mathbf{y}_\mathcal{U}|\mathbf{y})]$ is minimised.

Assume that $q(\mathbf{y}_\mathcal{U})$ is a normal distribution, i.e. assume that $q(\mathbf{y}_\mathcal{U}) \triangleq \mathcal{N}(\mu_\mathcal{U}, \Sigma_\mathcal{U})$ where the mean and the covariance function will be dependent on our choice of set $\mathcal{X}$. According to Hoang et al. (2015), it can be shown that the $\mu_\mathcal{U}$ and $\Sigma_\mathcal{U}$ which minimises the KL divergence will also maximise the ELBO as given by (Equation 6 of (Hoang et al., 2015))

$$L(q) = \int q(\mathbf{f}_\mathcal{U})\left[\int q(\mathbf{f}|\mathbf{f}_\mathcal{U}) \log \frac{p(\mathbf{f}, \mathbf{y}|\mathbf{y}_\mathcal{U})}{q(\mathbf{f}|\mathbf{f}_\mathcal{U})} d\mathbf{f}\right] d\mathbf{f}_\mathcal{U} - D_{KL}[q(\mathbf{f}_\mathcal{U})\|p(\mathbf{f}_\mathcal{U})] \tag{109}$$

where $\mathbf{f}$ is the true model function (i.e. the noiseless version of $\mathbf{y}$). In order to compute the distribution $q(\mathbf{y}_\mathcal{U})$ which maximises L, we can compute its gradient. In this case the gradient has a nice closed form, and is given by (Equation 17 of Hoang et al. (2015))

$$\frac{\partial L}{\partial \mu_\mathcal{U}} = \Theta_\mathcal{U}^{-1}\mu_\mathcal{U} - \sum_{(x,y)\in\mathcal{D}} \left[G(\mathcal{U}, x) - F(\mathcal{U}, x)\mu_\mathcal{U}\right] \tag{110}$$

and (Equation 18 of Hoang et al. (2015))

$$\frac{\partial L}{\partial \Sigma_\mathcal{U}} = \Sigma_\mathcal{U}^{-1} - \frac{1}{2}\Theta_\mathcal{U}^{-1} - \frac{1}{2}\sum_{x\in\mathcal{X}} F(\mathcal{U}, x) \tag{111}$$

where $G(\mathcal{U}, x) = \Theta_\mathcal{U}^{-1}\Theta_{\mathcal{U}x}\Gamma_x^{-1}y$, $F(\mathcal{U}, x) = \Theta_\mathcal{U}^{-1}\Theta_{\mathcal{U}x}\Gamma_x^{-1}\Theta_{x\mathcal{U}}\Theta_\mathcal{U}^{-1}\mu_\mathcal{U}$ and $\Gamma_x = \Theta_x - \Theta_{x\mathcal{U}}\Theta_\mathcal{U}^{-1}\Theta_{\mathcal{U}x}$. We will also let $\Theta_\mathcal{U} = \Theta(\mathcal{X}_\mathcal{U}, \mathcal{X}_\mathcal{U})$ and $\Theta_{x\mathcal{U}} = \Theta(x, \mathcal{X}_\mathcal{U})$. Using the closed form expression of the gradient, the optimal $\mu_\mathcal{U}$ and $\Sigma_\mathcal{U}$ can be computed through gradient ascent or by directly solving equations $\partial L/\partial\mu_\mathcal{U} = 0$ and $\partial L/\partial\Sigma_\mathcal{U} = 0$. In our experiments, we will use the latter method.

Given the optimal $\mu_\mathcal{U}$ and $\Sigma_\mathcal{U}$, the mean of the sparse NTKGP approximation can be computed as

$$\mu_{\text{sNTKGP}}(\mathbf{y}_T|\mathbf{y}) = \Theta_{T\mathcal{U}}\Theta_\mathcal{U}^{-1}\mu_\mathcal{U}^*, \tag{112}$$

and the covariance by

$$\Sigma_{\text{sNTKGP}}(\mathbf{y}_T|\mathbf{y}) = \Theta_T - \Theta_{T\mathcal{U}}\Theta_\mathcal{U}^{-1}\Theta_{\mathcal{U}T} + \Theta_{T\mathcal{U}}\Theta_\mathcal{U}^{-1}\Sigma_\mathcal{U}^*\Theta_\mathcal{U}^{-1}\Theta_{\mathcal{U}T}. \tag{113}$$

The approximate posterior can now be computed with time cubic with number of inducing points, rather than cubic in number of training points. The equation can also be iteratively computed in quadratic time if Eq. (104) is used for matrix inversion.

From the decomposition in Eqs. (110) and (111), we see that the FIC approximation is nice to use since it is possible to decompose the summation above based on terms $F(\mathcal{U}, x)$ and $G(\mathcal{U}, x)$ where each terms only depends on one element, and independent of the other elements in $\mathcal{D}$. This is convenient when trying to add one element and see what effect it has on the inducing points distribution. Using this fact, it is also possible to apply a further linear approximation on the criterion (as we will present in the next subsection) which allows for even more efficient computation.

C.3. Approximating incremental change of μ_{NTKGP} and Σ_{NTKGP}

Given the inducing point prior $\Sigma_\mathcal{U}(\mathcal{X})$ as computed by the method above, we could theoretically compute the approximate posterior $\Sigma_{\text{sNTKGP}}(\mathcal{X}_T|\mathcal{X} \cup \{x'\})$ when one point is added directly using the same method. However, this can still be expensive to perform. Therefore, we propose a simple technique to approximate the updated value of $\Sigma_\mathcal{U}(\mathcal{X} \cup \{x'\})$, and therefore $\Sigma_{\text{sNTKGP}}(\mathcal{X}_T|\mathcal{X} \cup \{x'\})$, when we add a small number of points to the training set. The approximation technique is akin to the influence function as proposed by Koh & Liang (2017).

When we add a single point to the training set, the ELBO (Eq. (109)) will only change by a small amount. Let $L' = L + \Delta L$ be the new loss function after adding x' into the set $\mathcal{X}$, and ΔL be the contribution specifically from the new element x'. Since L

only changes by an amount ΔL, we also do not expect $\Sigma_{\mathcal{U}}$ to change by much. Let $\Sigma_{\mathcal{U}}(\mathcal{X} \cup \{x'\}) = \Sigma_{\mathcal{U}}(\mathcal{X}) + \Delta\Sigma_{\mathcal{U}}(x'; \mathcal{X})$ where $\Delta\Sigma_{\mathcal{U}}(x'; \mathcal{X})$ is the change of the inducing prior after adding the training point x'. We see that $\Sigma_{\mathcal{U}}(\mathcal{X} \cup \{x'\})$ and $\Sigma_{\mathcal{U}}(\mathcal{X})$ would be slightly different from each other.

We find that the change in the inducing points posterior can be instead given by

$$\Delta\Sigma_{\mathcal{U}}(x'; \mathcal{X}) \approx -\left(\frac{\partial^2 L}{\partial \Sigma^2}\bigg|_{\Sigma=\Sigma_{\mathcal{U}}(\mathcal{X})}\right)^{-1}\left(\frac{\partial(\Delta L)}{\partial \Sigma}\bigg|_{\Sigma=\Sigma_{\mathcal{U}}(\mathcal{X})}\right) \tag{114}$$

Conveniently, the FIC approximation gives rise to a loss function whose gradient is easily decomposable. We can write the derivative of the loss function contribution from x' as

$$\frac{\partial(\Delta L)}{\partial \Sigma} = -\frac{1}{2}F(\mathcal{U}, x). \tag{115}$$

We also see that the Hessian of the original loss is equal to the Jacobian of the matrix inverse, i.e.

$$\frac{\partial^2 L}{\partial \Sigma^2} = \frac{\partial}{\partial \Sigma}\left(\frac{1}{2}\Sigma^{-1}\right) \tag{116}$$

which can easily be computed in closed form and also using any auto-differentiation package. Expressions from 115 and 116 can be plugged back into 114 to obtain the change of $\Sigma_{\mathcal{U}}$ when one new sample is added.

Therefore, any criterion α we have which would depend on the approximation of the sNTKGP can be approximated as

$$\Delta\alpha(x'; \mathcal{X}) \approx \left\langle \frac{\partial \alpha}{\partial \Sigma_{\mathcal{U}}}, \Delta\Sigma_{\mathcal{U}}(x'; \mathcal{X}) \right\rangle. \tag{117}$$

This expression can be used to compute the change in the active learning criterion when an additional data point is added. Because this term is based on the inner product of some matrix quantities, it is parallelisable and in practice speeds up the algorithm by a large amount.

Experimentally, the Hessian can be expensive to compute. In order to speed up the computation, we can instead use an alternate parametrisation of the sparse GP called the *natural parameters* which defines parameters to match more naturally to the terms that appears in the Gaussian distribution. We refer the readers to the original paper (Hoang et al., 2015) for further details on this.

D. Discussions On Active Learning Criteria

In this section, we discuss the active learning criteria based on the NTKGP which we use. We first provide further discussion on the expected value criterion, then we discuss two other possible criteria omitted from the main text based on variance percentile and on mutual information. The experiments comparing the performances of the other active learning criteria can be found in App. H.3.3.

D.1. Expected Variance Criterion

The expected variance criterion has been introduced in the main paper in (9). In this section, we would like to further explore how the active set is selected based on this criterion. In Fig. 8, we embed our data onto a space using t-SNE (van der Maaten & Hinton, 2008), then plot out the predictive variance at model initialization of different regions. An interesting observation from the plot is that the output variance with respect to model initialization is actually heteroscedastic. This means there will be some inputs which will have more varied model prediction than others at initialization. From the plot, we see that our algorithm does not necessarily pick the samples which evenly covers the whole input space, unlike what coverset-based algorithms may choose to do. Rather, our algorithm tends to also prioritise querying points from regions where the predictive variance is high, and balance it with querying from regions with a larger number of points.

D.1.1. Running Time

The computation of the criterion $\alpha_{\text{EV}}(\mathcal{X})$ has a cost of $\mathcal{O}(|\mathcal{X}_T| \cdot |\mathcal{X}|^2)$, where $\mathcal{X}_T$ is the reference test set. This is due to the fact that we have to compute the GP posterior variance for each of the element in $\mathcal{X}_T$. Computing the GP posterior is cubic

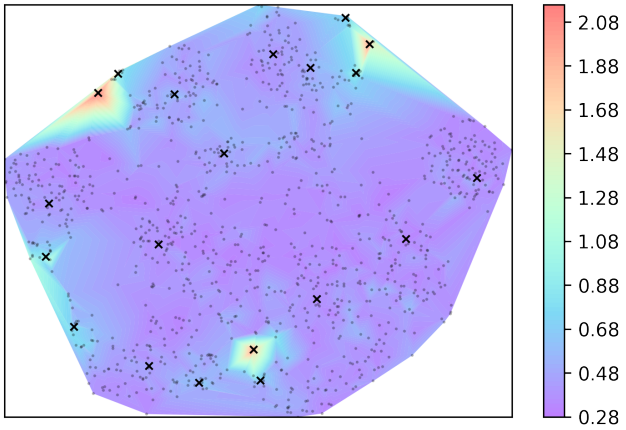

Figure 8. Visualisation of which points are selected by α_{EV} criterion. The black points represent the unlabelled set of data from the Handwritten Digits (from the UCI repository), mapped onto a 2D plane using t-SNE based on the predictive correlation (i.e. if the model output is more correlated, then they will be mapped closer in the above representation). The contours represent the predictive standard deviation of each input point, computed using the empirically with the model output at initialization. The crosses represent the points which are selected by our active learning algorithm using the α_{EV} criterion.

with respect to $\mathcal{X}|$, however can be made quadratic due to the trick of incrementally computing the NTKGP which we will discuss in App. C.1. Note that since the NTK is assumed to be constant from initialization, it can be precomputed at the beginning and reused throughout the active learning process, incurring little cost during the actual active learning process.

D.1.2. Practical Speedups

In additional to the speedups in computations as discussed in App. C.2, we also make further approximations in order to compute α_{EV}. Namely, rather than computing the empirical mean of output variance on the whole test set $\mathcal{X}_T$, we first select a subset of the test set using simple diversification methods (e.g. K-Means++), then only compute the criterion on those data. We find that not only does this speed up the criterion computation, this technique also remove some bias in the test data distribution and give a more representative of the overall variance across the whole space. We also optimize our computation of α_{EV} further by only computing the variance σ^2_{NTKGP} and not the whole covariance Σ_{NTKGP}.

D.2. Variance Percentile

Another possible idea is to ensure that some proportion of test points has a low variance. This can be done by considering the rth percentile of variance, i.e.

$$\alpha_{\text{rV}}(\mathcal{X}) = -\text{ percentile}\big(\{\sigma^2_{\text{NTKGP}}(x|\mathcal{X}) | x \in \mathcal{X}_T\}, r\big). \tag{118}$$

In this case, letting $r = 50$ is equivalent to considering the median of test output variance, and letting $r = 100$ is equivalent to considering the maximum test output variance.

Unfortunately, the maximum function is not submodular in general, and therefore has no theoretical guarantees when points are selected using the greedy algorithm. Nonetheless, we will conduct experiments to select points based on this criterion for the cases where $r = 90$ and $r = 100$.

D.3. Mutual Information Criterion

The previous criteria we have discussed often does not take into account the distribution of $\mathcal{X}_T$. Through the lens of information theory, we can view the active learning problem as attempting to select points $\mathcal{X}_\mathcal{L}$ with outputs $\mathbf{y}_\mathcal{L}$ such that the most information can be obtained about $\mathbf{y}_T$, and that the dependence between each datapoint is also taken into account.

Similar to Krause et al. (2008), we can attempt to maximise the mutual information

$$\alpha_{\text{MI}}(\mathcal{X}_{\mathcal{L}}) = \mathbb{I}[\mathbf{y}_{\mathcal{L}}; \mathbf{y}_{T\setminus\mathcal{L}}] \tag{119}$$
$$= \mathbb{H}[\mathbf{y}_{T\setminus\mathcal{L}}] - \mathbb{H}[\mathbf{y}_{T\setminus\mathcal{L}}|\mathbf{y}_{\mathcal{L}}] \tag{120}$$
$$= \frac{1}{2}\log\det\Sigma_{\text{NTKGP}}(\mathcal{X}_{T\setminus\mathcal{L}}) - \frac{1}{2}\log\det\Sigma_{\text{NTKGP}}(\mathcal{X}_{T\setminus\mathcal{L}}|\mathcal{X}_{\mathcal{L}}) + \text{constant}, \tag{121}$$

where we use the shorthand $\mathcal{X}_{T\setminus\mathcal{L}} = \mathcal{X}_T \setminus \mathcal{X}_{\mathcal{L}}$. We are able to arrive at the final line since we know the predictive covariance follows a multivariate Gaussian distribution.

In the case that the test set and the unlabelled pool are disjoint, we have $\mathbf{y}_{T\setminus\mathcal{L}} = \mathbf{y}_T$, which means $\mathbb{H}[\mathbf{y}_{T\setminus\mathcal{L}}]$ is a constant regardless of which active set we choose. In this case, maximising α_{MI} is equivalent to minimising $\mathbb{H}[\mathbf{y}_T|\mathbf{y}_{\mathcal{L}}]$. Similar to the case of α_{EV}, in experiments where the test set is separate from the unlabelled pool, we will simply report the values of $-\mathbb{H}[\mathbf{y}_T|\mathbf{y}_{\mathcal{L}}]$.

D.3.1. Numerical Stability

A particular issue with using the mutual information is the numerical stability when computing the entropy values. In particular, α_{MI} involves computing the entropy $\mathbb{H}[\mathbf{y}_{T\setminus\mathcal{L}}|\mathbf{y}_{\mathcal{L}}]$, which involves computing the determinant $\det\Sigma_{\text{NN}}(\mathcal{X}_{T\setminus\mathcal{L}}|\mathcal{X}_{\mathcal{L}})$. However, when the test set $\mathcal{X}_{T\setminus\mathcal{L}}$ have points which are highly correlated with each other, the matrix $\Sigma_{\text{NN}}(\mathcal{X}_{T\setminus\mathcal{L}}|\mathcal{X}_{\mathcal{L}})$ may be singular and cause the determinant to be undefined.

Notice that the decision of using $\mathcal{X}_{T\setminus\mathcal{L}}$ instead of $\mathcal{X}_T$ will already alleviate some of the issues regarding singular matrices. However, in order to prevent further issues, in our experiments, we decide to pre-filter the points that are used for the test set. In particular, instead of using all points from the test set, we select only a subset of training points such that $\mathbb{H}[\mathbf{y}_T]$ is maximised. This means that the subset selected will be as independent from each other as possible. Furthermore, using a subset of points instead of the full test set also reduces the matrix size, which speeds up our computation as well. The subset of points that are selected are done so using the K-Means++ initialization method.

Furthermore, to avoid computing the determinant of a singular matrix, we add in a diagonal noise term in order to compute the determinant of $\Sigma_{\text{NN}}(\mathcal{X}_{T\setminus\mathcal{L}}|\mathcal{X}_{\mathcal{L}}) + \sigma_n^2 I$ instead. This corresponds to the case where the observation has some added noise with variance σ_n^2.

E. Further Details about the Greedy Algorithm

As denoted in Algo. 1, for the data selection stage, we utilise the simple greedy algorithm to find the set of suitable points. However, the greedy algorithm requires $\mathcal{O}(nk)$ criterion computations, which can be slow.

There is a large literature on efficient submodular maximisation algorithms with cardinality constraints. This, however is not the focus of this paper. Furthermore, since the data selection stage is independent of the true labels, our algorithm can Nonetheless, for practical purposes, we choose to apply two simple algorithm optimization techniques.

The first optimization performed based on the Accelerated-Greedy algorithm (Minoux, 1978). For each element in the unlabelled data pool, we store the marginal gain value $\Delta_x(\mathcal{X}) \triangleq \alpha_{\text{EV}}(\mathcal{X} \cup \{x\}) - \alpha_{\text{EV}}(\mathcal{X})$ for some $\mathcal{X}$ that was previously computed. Then, as the greedy algorithm progresses, we will continue to grow the selected active set into some $\mathcal{X}' \supset \mathcal{X}$. From submodularity, we know that $\Delta_x(\mathcal{X}) \geq \Delta_x(\mathcal{X}')$. This means that in one particular round with set $\mathcal{X}'$, if we already have checked another point whose marginal gain in that round is such that $\Delta_{x'}(\mathcal{X}') \geq \Delta_x(\mathcal{X})$, then there is no point in checking x' since we know the criterion value of $\mathcal{X}' \cup \{x'\}$ will definitely be higher than that obtained from $\mathcal{X}' \cup \{x'\}$. In our algorithm, when the empirical NTK is used, we reinitialize the random neural network after each batch of data (which changes the empirical NTK), and as a result also reset the stored values of Δ_x.

Another optimization used is that in each greedy round, we do not compute the criterion on all elements, but only on a subset of k' elements per round. In each greedy round, the maximum value amongst the k' elements tested is added to the active set. This is the technique used in the Stochastic-Greedy algorithm (Mirzasoleiman et al., 2015), and is able to achieve accuracy arbitrarily close to $1 - 1/e$ in $\mathcal{O}(n)$ time (independent of k). In our experiments, we either set $k' = 1000$ for experiments with smaller budgets and $k' = 250$ for experiments with larger budgets.

F. Proof of Eq. (10)

We verify (10) for MSE loss. For a single test point (x, y), notice that we can write

$$\begin{aligned}\mathbb{E}_{\theta_0}\left[\ell\big((x,y),\text{train}(\theta_0)\big)\right] &= \mathbb{E}_{y'\sim\mathcal{N}(\mu_{\text{NN}}(x|\mathcal{X}),\sigma^2_{\text{NN}}(x|\mathcal{X}))}\left[(y'-y)^2\right] \\ &= \mathbb{E}_{y'}[(y')^2] - 2y\mathbb{E}_{y'}[y'] + y^2 \\ &= \mu_{\text{NN}}(x|\mathcal{X})^2 + \sigma^2_{\text{NN}}(x|\mathcal{X}) - 2y\mu_{\text{NN}}(x|\mathcal{X}) - y^2 \\ &= (y-\mu_{\text{NN}}(x|\mathcal{X}))^2 + \sigma^2_{\text{NN}}(x|\mathcal{X})\end{aligned}$$

where the first equality is based on the predictive distribution of the converged model.

G. Further Details on Experimental Setup

In this section, we provide further details of the experiments conducted. All the code for the experiments are also provided in the supplementary material.

G.1. Training Setup for Each Experiments

G.1.1. Regression Experiments

We will use a combination of generated dataset and real-life data. For this paper, randomly generated data (Random Model) is constructed from random points from a ball, with their output values generated from a random model initialization (which ensures that the neural network will be adequate to describe the data used).

Meanwhile the real-life training data are taken from the UCI Machine Learning Repository (Dua & Graff). For all dataset, we split the whole data in half, and let one half be the pool of data and the other be the test data which the algorithm has no access to. All the dataset used are regression datasets, with the exception of the Handwritten Digits dataset which is a classification dataset where we perform regression on the label value (i.e. for the inputs corresponding to the digit 1, we assign $y = 1$). All datasets are also normalized such that they have mean of zero and variance of 1.

In all of the regression experiments, the model used is a two-layer multi-layer perceptron with width of 512 and with bias. We set $\sigma_W = 1.$ and $\sigma_b = 0.1$. The NNs are optimized using gradient descent with step size 0.01.

G.1.2. Experiments on the Classification Dataset

In the classification datasets, in order to reduce the training time, we restrict the unlabelled data pool to a random subset of the whole training set. For the MNIST dataset, we randomly select 10k points and use it as our unlabelled pool, while for the remaining classification experiments we randomly select 20k points for the unlabelled pool. All the inputs are also rescaled such that the distribution of input values are between $[-1, 1]$. For all the models, we train the models using stochastic gradient descent with learning rate of 0.1 and weight decay of 0.005. The models are trained with training batch size of 32 and are trained for 100 epochs. Below, we provide a brief description of the datasets used and the model architectures used for training on the correpsonding datasets.

- **MNIST** (Deng, 2012). MNIST is a database whose inputs are single-channel images of size 28×28 that corresponds to handwriting of different digits. For MNIST, we use a multi-layer perceptron with two hidden layers each of width 512 as we have done for the regression experiments. We find that using such a simple model without convolution is sufficient for the MNIST dataset. In the experiments where we vary the network width at active learning stage and training stage (Fig. 21), we used a 3-layer MLP and experiment with network widths 128, 256, 512 and 1024. The total query budget was fixed at 600 and the query size at 200.
- **EMNIST** (Cohen et al., 2017). The EMNIST dataset is an extension to the MNIST dataset, where the input contains handwritten characters as well as handwritten digits. There are a total of 47 output classes (since some of the characters that have a similar shape are grouped together in the same group). For EMNIST, we either use a multi-layer perceptron with three hidden layers each of width 512, or a simple convolutional neural network with 2 blocks of [Conv - MaxPool - ReLU], followed by a fully-connected layer of width 512.
- **SVHN** (Netzer et al., 2011). SVHN consists of RGB images of size 32×32 which corresponds to photos different digits. Each image in the SVHN dataset corresponds to one of the 10 classes, each representing a digit. For SVHN,

we use a 1-layer WideResNet (Zagoruyko & Komodakis, 2016). We modify the network by removing the batch normalization layer in order to make the module compatible with FUNCTOOLS on Pytorch (see App. G.2) such that the NTK is computable by gradient inner product, and also to ensure positive-semidefiniteness of the resulting matrix. While this may result in a lower model accuracy, this is not a main concern of this work regardless since we are more interested in the *relative* accuracy between each active learning methods.

- **CIFAR100** (Krizhevsky, 2009). CIFAR100 is a dataset consisting of RGB images of size 32×32 which corresponds to one of 100 different image categories. For CIFAR100, we use a 2-layer WideResNet with similar modifications as we have done for SVHN.

G.1.3. MODEL SELECTION ALGORITHM

For the model selection experiments, the data used are from the UCI dataset (as we have done for the regression experiments). We also construct an artificial dataset Sinusoidal whose output is generated from the sine function $y = \sin(\omega^\top x + b)$ for some random ω and b.

The pool of models $\mathcal{M}$ consists of MLPs with depths 1, 2, 3 and 4, and with ReLU, GeLU, leaky ReLU ($a = 0.1$), and Erf. This makes up a total of 16 candidate models. For the Sinusoidal dataset, we also added an additional model architecture with sinusoidal activation, making a total of 20 candidate models in this case. All the models are implemented on JAX (in order to match the theoretical NTK which is also computed in JAX). For the model selection algorithm, in order to approximate α_M, we choose $\kappa = 0.3|\mathcal{L}|$ for the labelled set $\mathcal{L}$ at that round, and compute the empirical mean of α_M by sampling the random subset 100 times. The batch size for active learning is 10 or 20 depending on the size of the dataset.

The models are trained using the Adam optimizer with learning rate 0.01. The reason the Adam optimizer is used is that some of the activation functions suffer from diminishing gradients, and therefore does not behave well when using gradient descent. The training is all done in a single batch, and is done for 500 epochs.

G.2. Computation of the Neural Tangent Kernel

G.2.1. THEORETICAL NTK USING NEURAL-TANGENTS

The theoretical NTK are computed using the NEURAL-TANGENTS package (Novak et al., 2019). Even though the final trained model will not have exactly the same network parameters as that assumed to compute the theoretical NTK (due to different packages used during active learning and true model training), we still find that the kernel itself shows enough agreements and are still useful for predicting the model behaviour.

The main disadvantage of using the theoretical NTK is that the theoretical NTK is not available for all model architectures. In particular, NEURAL-TANGENTS does not work for MaxPooling layers (only AvgPooling is supported), and also does not work for BatchNorm layers or Dropout layers placed arbitrarily. For this reason, experiments involving the theoretical NTK are only restricted to ones which uses a MLP. We also only use the theoretical NTK for experiments involving our algorithms, and do not use it for MLMOC which uses the empirical kernel in order to predict how a specific model initialization will behave during training.

G.2.2. EMPIRICAL NTK USING PYTORCH

An alternative method of computing the NTK is to do so empirically. In order to compute the NTK empirically, we require taking the inner product of the model output gradient $\nabla_\theta f(x, \mathcal{X})$ with respect to its parameters. This requires us to compute the Jacobian $\nabla_\theta f(x, \mathcal{X})$, which is extremely expensive in terms of required memory and cost for performing the matrix multiplication. In particular, for a model with p parameters and o outputs, computing the NTK on an input of size n requires $\mathcal{O}(npo)$ space and $\mathcal{O}(n^2po)$ running time for the matrix multiplication alone. In order to perform this operation efficiently, we follow the method as used in PyTorch's FUNCTOOLS tutorial[5]. Note that FUNCTOOLS is not compatible with all neural network components (e.g. BatchNorm), and therefore in our experiments we do not use those components in the models. Even though `Neural-Tangents` is able to compute the empirical NTK as well, we chose to compute the empirical NTK on PyTorch since the models and (most of) the model training are done on PyTorch anyway.

For classification instances where the model has multiple outputs, we also perform a further approximation of only using the gradient which contributes to a single model output (which is chosen at random). This is possible since the gradient inner

[5]See `https://pytorch.org/functorch/stable/notebooks/neural_tangent_kernels.html`.

product with respect to each outputs are independent of each other and has the same value in the limit. This is a similar trick which is also used by Mohamadi et al. (2022).

G.3. Reported Metrics

To quantify the model performances, we use either the test mean-squared error (MSE) for regression problems, or the test accuracy for classification problems.

To quantify the initialization robustness of the models, we use two different metrics as described below.

- **Output variance**. This is used as an initialization-robustness measure for regression tasks, and is the empirical variance of output of trained neural networks with respect to the different model initializations. In our experiments, we report the 90th percentile output variance, which is defined as the 90th percentile output variance test data. The 90th percentile is used instead of the 100th percentile (i.e. the maximum output variance) since using the 100th percentile tends to focus on some outlier rather than reflecting the output variance of the majority of the test data. We also chose to report the 90th percentile value instead of the average value since it is able to indicate the output variance of the worst-case inputs and is therefore a better measure of initialization robustness on the overall space.
- **Output entropy**. This is used as an initialization-robustness measure for classification tasks, and is defined as the empirical entropy of the predicted label. For a neural network, the predicted label for some input x is given by $\hat{y} = \arg\max_i f(x;\theta)_i$. Given multiple trained models with different parameters $\theta_1, \ldots, \theta_k$, we will obtain different predictions $\hat{y}_1, \ldots, \hat{y}_k$. The output entropy is then defined as $-\sum_{i=1}^{c} v_i \log v_i$ where $v_i = \frac{1}{c} \cdot \sum_{j=1}^{k} \mathbb{1}[\hat{y}_j = i]$. A lower output entropy corresponds to models whose predictions are more consistent with each other.

G.4. Description of Other Active Learning Algorithms

Below, we provide a brief description of the other active learning algorithms which we compare our method against.

- RANDOM. The batch of points are selected uniformly at random without replacement.
- K-MEANS++. The batch of points are selected based on the initialization method from Arthur & Vassilvitskii (2007). When using this algorithm, all the points are selected right from the start (and order in selection by the algorithm is kept). When the user queries for a batch of size b, the algorithm returns the next b elements that it has chosen from the K-MEANS++ initialization algorithm.
- BADGE (Ash et al., 2020). BADGE utilizes the hallucinated gradient space, or the gradient of the loss function with respect to the final output layer,

  $$h(x) = \frac{\partial}{\partial \theta^{(-1)}} \ell((x, \tilde{y}), \theta)$$

  where $\theta^{(-1)}$ are the model parameters of the final output layer and $\tilde{y}$ is the pseudo-prediction from the model based on the model output (for example, in classification problems, $\tilde{y}$ would be the one-hot vector representing the class prediction given by the model output). Pseudo-labels are used since in active learning, the algorithm would not have access to the true labels. Given the embedding $h(x)$, BADGE then performs diversification on this embedded input. Theoretically, BADGE is able to provide a balance between selecting points which are diverse and selecting points which the model is uncertain about.

  Unlike in the original paper, for experiments involving BADGE, we do not provide the algorithm with an initial training set. Despite this, we find that BADGE is still able to select a good batch of points from the start, and is even able to beat our algorithm in some experiments.
- MLMOC (Mohamadi et al., 2022). MLMOC uses the NTK to provide a prediction of how a neural network would behave when an additional point is added to the training set. Given this, the algorithm selects an active set which when trained, will cause the largest change in the model output.

  In our experiments, MLMOC will randomly select 100 initial points for classification experiments. The random subset is counted as a part of the budget used up by MLMOC. For SVHN and CIFAR100 experiments, this is fewer points than provided in the original paper. However, this setup is chosen in order to test how the algorithm performs when there is little initial data available. We found that MLMOC performs poorer when there are fewer initial data points available to it.

H. Further Details on Experimental Results

H.1. Description of Figures

In this section, we give a more detailed description of the figures plotted in the main paper.

- **Fig. 2**. This figure shows the correlation between variance predicted by the NTKGP, $\sigma^2_{\text{NTKGP}}(x|\mathcal{X})$, and the (empirical) model output variance. The horizontal represents the variance based on $\sigma^2_{\text{NTKGP}}(x|\mathcal{X})$ for a certain element x from the validation set and a certain subset $\mathcal{X}$ of the unlabelled data pool, while the vertical represents the empirical output variance with respect to different model initialization. In this experiment the Robot Kinematics dataset was used.

- **Fig. 3**. This figure shows the relationship between the different criteria value, the 90th percentile output variance, and the test MSE. In each row, from left to right, the graph shows: the relationship between the 90th percentile variance and α_{EV}; the 90th percentile output variance of models when trained on active sets of increasing sizes; the test MSE when trained on active sets of increasing sizes. The blue line represents the active set which has been sequentially constructed based on the α_{EV} criterion. Each row represents trials conducted on one dataset. SP and PR represents the Spearman Rank correlation and Pearson correlation coefficients respectively. The metrics reported on the y-axis are described further in App. G.3.

H.2. Relationship Between σ_{NTKGP}, σ_{sNTKGP} and Output Variance

In Fig. 9, we provide further results comparing the variance obtained from the σ_{NTKGP}, σ_{sNTKGP} approximations and the true output variance. We find that σ_{NTKGP} provides good agreement with the output variance, while σ_{sNTKGP} provides a decent approximation of the true output variance. Furthermore, note that the graph shows the *individual* variance and not the sum of variance (as α_{EV} requires) so when computing the criterion, the discrepancy between using the NTKGP or sNTKGP approximation and using true output variance will be lower (see Fig. 10).

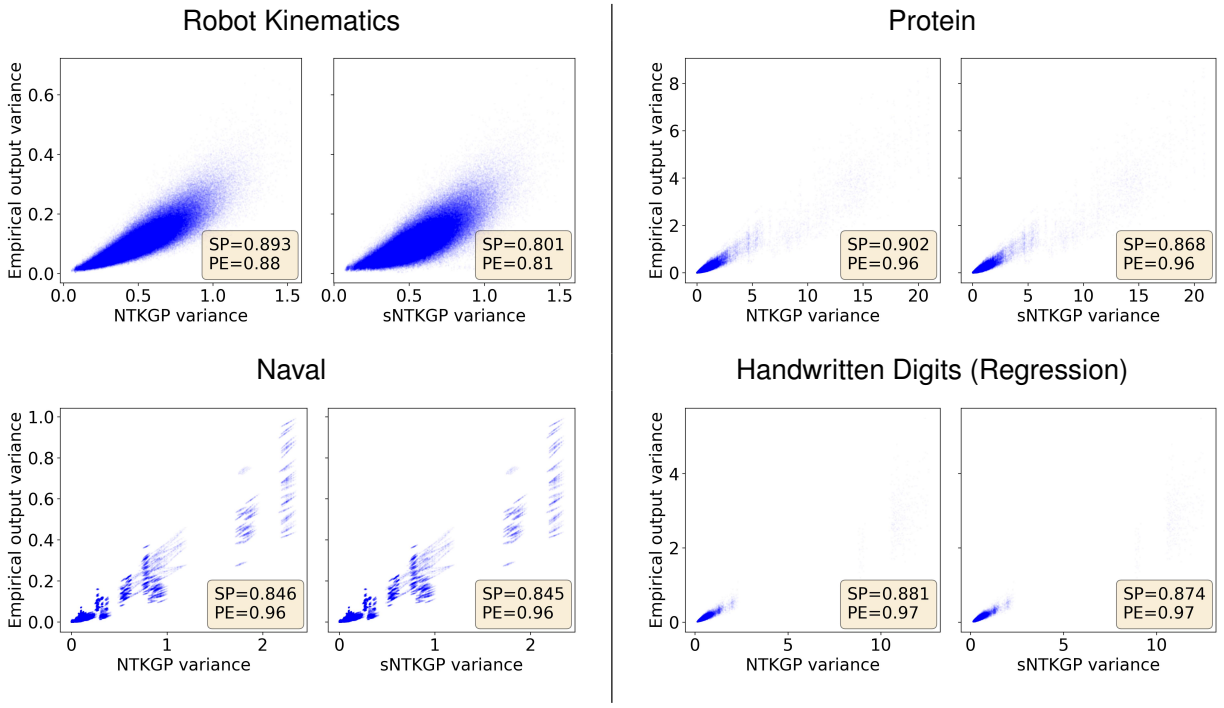

Figure 9. The correlation between variance predicted by the NTKGP and sNTKGP, and the (empirical) model output variance. The x-axis represents the variance based on the NTKGP or the sNTKGP for a certain element x from the validation set and a certain subset $\mathcal{X}$ of the unlabelled data pool, while the y-axis represents the empirical output variance with respect to different model initialization.

H.3. Additional Results From Regression Experiments

H.3.1. Experiments on Other Regression Datasets

In Fig. 10 we show further results for other regression datasets in the sequential selection process, and in Fig. 11, we show the results in the larger-scale, batch setting.

We see that EV-GP criterion is able to outperform both the Random selection and K-Means++ selection method. The latter result is true likely due to the fact that K-Means++ aims at selecting points which are as diverse as possible from each other in the input space. This, however, makes K-Means++ more prone to selecting outliers which have large differences with other points in the input space yet have little correlation with the rest of the dataset. Therefore, these outliers would provide little information for making predictions on the more "common" data points in the test set. Meanwhile, EV-GP criterion directly incorporates the distribution of the dataset in the point selection process, and therefore is able to select more correlated points based on the existing dataset.

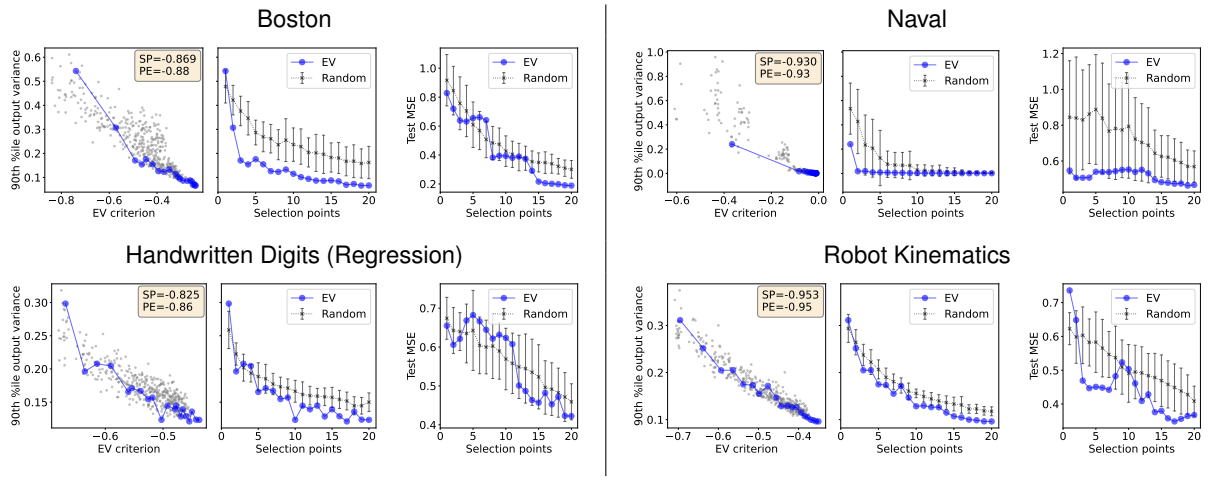

Figure 10. Relationship between the different criteria value, the 90th percentile output variance, and the test MSE. The information shown is presented in the same manner as Fig. 3 (see App. H.1).

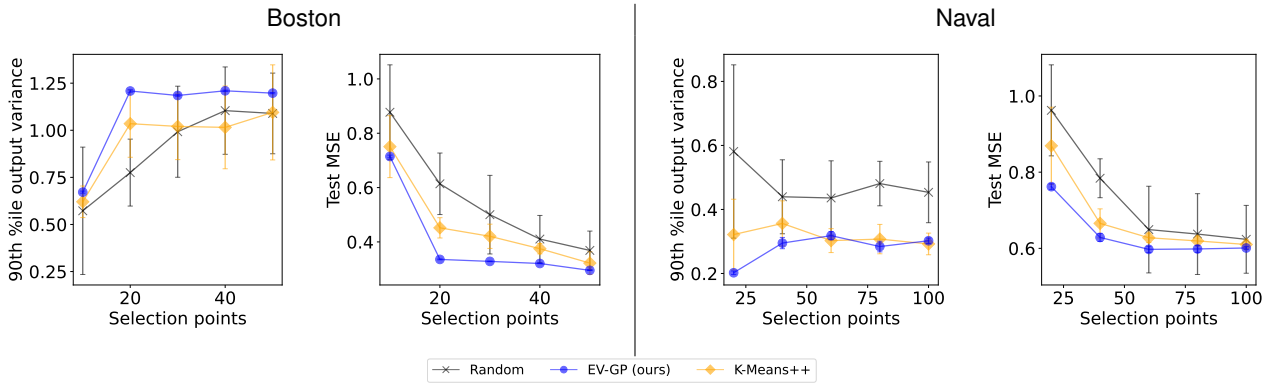

Figure 11. Active learning on regression datasets. The information shown is presented in the same manner as Fig. 4 (see App. H.1).

H.3.2. Suitability of α_{EV} and MSE Loss

In Fig. 12, we study how the ability of α_{EV} to represent the test loss is able to reflect how suitable the model is for the given dataset. We find that when α_{EV} shows a higher correlation to the test loss, the achievable test loss (when the model is trained on the entire set) is also lower. This shows that if we have a model which matches closely to the true data, then optimizing α_{EV} is more likely to result in a low test loss. This fact can be related back to Theorem B.1, where we have the constant B which represents how well the underlying function fits with the neural network model. The lower value of B means that

optimizing α_{EV} can provide a tighter bound on the difference between the trained model and the true function. We also note that despite the results showing low correlations between α_{EV} and MSE loss in some of the datasets, we are still able to minimize the loss simply by maximising α_{EV}.

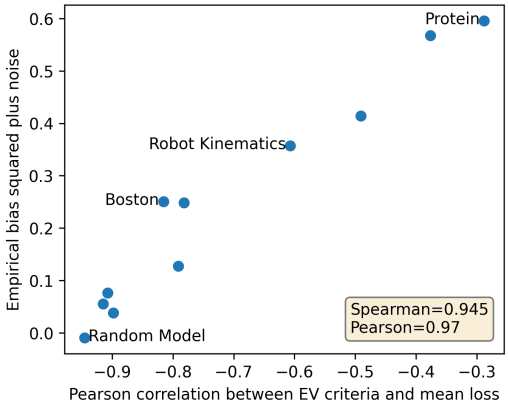

Figure 12. Plot between the correlation of $\alpha_{\text{EV}}(\mathcal{X})$ and the MSE loss when trained on $\mathcal{X}$, and the empirical bias (defined as the average MSE loss minus the output variance), on different regression datasets. Note that α_{EV} is higher when the output variance and the MSE loss is *lower*, meaning the correlation values on the x-axis are all negative. The Random Model dataset is a dataset whose outputs are generated from a randomly initialized 2-layer MLP (which the neural network is expected to fit well on). The results for active learning on the datasets labelled in the graph are shown in Fig. 13.

H.3.3. Results for Other Active Learning Criteria

We present some results for other active learning criteria presented in App. D. In Fig. 14 we show the results in the sequential setting, and in Fig. 15 we show the results in the batch setting.

We find that $\alpha_{100\text{V}}$ tends to perform poorly compared to the other criteria, and sometimes even worse than Random in some instances (in Fig. 15 we did not show some of the results from $\alpha_{100\text{V}}$ since it gave a training set which was unstable during training). We believe that this is due to the fact that $\alpha_{100\text{V}}$ will sometimes tend to be too focused on reducing the variance of outliers that it does not provide a diverse subset of points. Reducing this threshold to the 90th percentile (and using $\alpha_{90\text{V}}$) is able to give a better subset however still is not as effective as α_{EV}.

Meanwhile, α_{MI} often results in good performances, and sometimes even outperforming α_{EV}. However, we find it is still less preferable since its advantage over α_{EV} is not consistent and also due to the aforementioned issue of computational efficiency.

H.3.4. Results for Approximation Methods of NTKGP

In Figs. 16 and 17, we present the results for active learning results for various approximation methods of the NTKGP, namely approximating the NTK empirically, and approximating the covariance using sparse GP techniques (as introduced in App. C.2).

We can see that the empirical approximation of the NTK sees some drop in performance compared to the theoretical NTK, however still provides useful enough approximation for the true NTK for our purposes.

Meanwhile, in the experiments where the covariance from the sNTKGP is used, we can see that the method also still provides a useful approximation for the criterion, and may be used instead of the true NTKGP, at least for smaller problems. Unfortunately, we find that applying a further linear approximation to the sNTKGP computation worsens the selection process. This suggests that the covariance function from the sNTKGP may not be smooth enough and so assuming a linear approximation method is inaccurate.

In practice, we also find that for smaller datasets (where the number of selected points is not smaller than the number of inducing points), the sNTKGP approximation is not as beneficial since the computation required for the full-rank NTKGP is already low and sNTKGP requires more matrix computation that would otherwise. Nonetheless, the sNTKGP may be

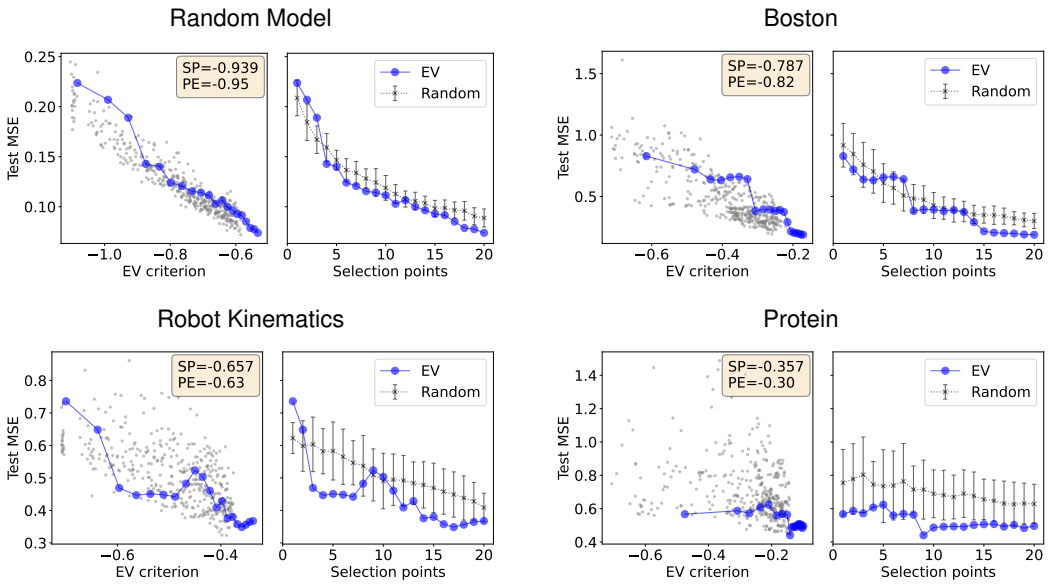

Figure 13. Plot comparing the values of $\alpha_{\text{EV}}(\mathcal{X})$ and the MSE loss when trained on $\mathcal{X}$ on different regression datasets. For each dataset, the left plot shows $\alpha_{\text{EV}}(\mathcal{X})$ and test MSE when trained on different subsets of $\mathcal{X}$, and the right plot shows the test MSE when trained on subsets of different sizes. The blue line shows the subset selected by our greedy algorithm to maximize α_{EV}. The datasets in the top row shows higher correlation between the criterion and the corresponding MSE loss, whereas the bottom row shows the opposite case. This correlation also reflects how well the chosen model architecture fits with the dataset.

a useful approximation method if can be scaled, and can be an interesting future direction of making our method more scalable.

H.4. Additional Results From Classification Experiments

H.4.1. Additional Results From Experiments Involving Neural Networks with Convolutions

In Fig. 18(a-b), we show the corresponding output entropy of the experiments from Fig. 5(c-d). We find that even for convolution experiments our method can generally select points which result in good initialization-robustness. In Fig. 18(c) we provide results for active learning on SVHN dataset on ResNet18 (following the setup from (Ash et al., 2020)), while Fig. 18(d-e) present the results for active learning on CIFAR100 dataset.

H.4.2. Comparison between EV-GP and BatchBALD

We also conducted experiments to compare EV-GP with BatchBALD (Kirsch et al., 2019), which is popular uncertainty-based active learning algorithm. BatchBALD was not included in the original benchmark due to its

In Fig. 19, we present the results of the experiments. In these experiments, we find that EV-GP outperforms BatchBALD, which shows that BatchBALD is less suitable for scenarios with no initial training data or when large batch sizes are used (which is a use case not targeted at and hence not tested by Kirsch et al. (2019)). BatchBALD also does not take into consideration the overall distribution of the training dataset which may also harm its performances.

H.4.3. Additional Results From Varying Active Learning Batch Sizes

In Fig. 20, we present further results for neural network active learning when the batch sizes vary. We find our algorithm shows little change in both accuracy and output entropy as the batch size gets larger. Note that when the empirical NTK is used, the algorithm seems to perform slightly worse in larger batch sizes case. This may be due to the chance that in some randomized network the empirical NTK provides a slightly less accurate representation of the true uncertainty. In the smaller iterations the random neural network used for NTK computation is reinitialized and so such an error can be reduced. However we find that in practice this effect is less significant and we still achieve a good active set regardless.

Table 1. Running time for experiments on MNIST dataset on 2-layer MLPs (corresponding to experiments in Fig. 5(a)).

Algorithm	Time to query n points (in seconds)			
	$n = 100$	$n = 200$	$n = 300$	$n = 400$
BADGE	12 ± 1	23 ± 1	35 ± 2	47 ± 2
BATCHBALD	2629 ± 1570	4499 ± 1571	6371 ± 1572	8227 ± 1574
MLMOC	88 ± 11	441 ± 62	974 ± 98	1637 ± 120
EV-GP	1120 ± 9	2089 ± 12	3159 ± 33	4283 ± 41

Table 2. Running time for the experiments using ResNet18 on the SVHN dataset (corresponding to experiments in Fig. 18(c)).

Algorithm	Time to query n points (in seconds)				
	$n = 200$	$n = 400$	$n = 600$	$n = 800$	$n = 1000$
BADGE	34 ± 4	70 ± 8	105 ± 9	148 ± 13	204 ± 15
MLMOC	2486 ± 3	5485 ± 38	9816 ± 43	16564 ± 62	26607 ± 457
EV-GP	3768 ± 10	7846 ± 10	13020 ± 52	18755 ± 91	25018 ± 115

H.4.4. Discussion on Running Time of Active Learning Algorithms

Although the active set selection process can incur long running time, in the practical deployment of active learning, the time required for active set selection is easily dominated by the costs of querying data labels and training the neural networks. To this end, EV-GP is able to reduce the *practical* running time in two aspects. Firstly, since active learning is usually required when querying data labels incurs high costs (e.g., costs in terms of time), an active learning algorithm which requires a smaller number of query batches (while ensuring good performances), is less time-consuming in practice. Secondly, significant source of the time costs in active learning results from the training of the neural networks, which is further aggravated in active learning problems involving a large amount of data. Since EV-GP is both batch size independent (as presented in Fig. 6) and training-free, EV-GP can reduce the practical costs in these aspects.

Regardless, we may still analyze the amount of time our active learning algorithm requires in performing active set selection. We present the running time of our experiments, which does not account for the time costs of querying data labels. Specifically, in Table 1, we present the running time for active set selection on MNIST dataset on 2-layer MLP, which corresponds to experiments in Fig. 5(a). Meanwhile, in Table 2, we present the running time for active set selection on SVHN dataset on ResNet18, which corresponds to experiments in Fig. 18(c)). These results show that the running time of EV-GP is not significantly larger than that of other algorithms which utilizes the NTK (e.g., MLMOC) or other sequential greedy-based active learning algorithms (e.g., BATCHBALD). Therefore, our EV-GP outperforms the other algorithms in terms of both the predictive performances and initialization robustness (as shown in our previous results) without incurring significantly more running time. As a consequence, in practical settings where the cost of labelling is higher, we expect EV-GP to have comparable or better time efficiency than other active learning algorithms due to its ability to run in large batches with little sacrifice to its performances.

H.4.5. Results From Varying Neural Network Width

Here we investigate how the width of the NN affects the performance of our EV-GP-EMP algorithm, by changing the widths of both the NNs used for data selection (i.e., for calculating our EV-GP criterion (Sec. 3.3)) and for training. The left figure in Fig. 21 shows that increasing the width of either the NN for data selection (horizontal axis) or for training (different lines) improves the initialization robustness This is because a wider NN for both data selection and training can reduce the error between the empirical NTK and exact NTK (Sec. 2.1), and therefore improve the accuracy of approximation for σ^2_{NTKGP} (6) and our EV-GP criterion (9). As a result, this leads to more accurate data selection and hence better initialization robustness. The right figure in Fig. 21 shows that increasing the width of the NN for training improves test accuracy, which can be attributed to the better expressivity of wider NNs (for the fixed width). Meanwhile, in contrast to initialization robustness, increasing the width of the NN for data selection does not have a significant impact on the test accuracy. This suggests that the NN does not need to be extremely wide in order to select data points that lead to a good predictive performance of the trained NN.

H.4.6. Effect of Initial Points on Other Active Learning Benchmarks

For BADGE and MLMOC algorithms, we find that the model performance is sometimes affected by how many initial labelled points were given to the algorithms. We plot the algorithms when some random data is available versus the case where no random initial data is given in Fig. 22.

We find that BADGE shows little difference whether the first batch of data is randomized or not. This demonstrates the BADGE is able to select diverse points by itself without needing extra information to kickstart the algorithm. MLMOC, on the other hand, seems to be more reliant on the initial dataset, with a noticeable drop in performance when the algorithm is given no initial data. This suggests that MLMOC is a poor choice in the low-data regime, when the active learning algorithm has to provide good performance right away even when little prior knowledge about the data is given.

H.5. Additional Results for the Model Selection Algorithm

H.5.1. Relationship Between α_M and True Loss

In this section we present how our model selection criterion relates to the true MSE loss. In Fig. 23, we present how the model selection criterion α_M relates to the true achievable MSE loss. Note that we do not expect α_M to be a perfect reflection of the test MSE since during the active learning we do not know what the true test set is, and that α_M will be dependent on what $\mathcal{L}$ is at the current point. Regardless, we find that α_M is able to give an accurate reflection of the MSE loss achievable.

H.5.2. Additional Results on Other Regression Datasets

In Fig. 24, we present further results for EV-GP+MS on other regression datasets. We find that in most datasets, our algorithm is able to select a suitable model architecture for the dataset, which leads to a low MSE loss at test time.

An issue we have found is that our algorithm is often sensitive to the obtained value of α_M. In some cases, multiple models will have a similar value of α_M, but in practice will have different MSE loss at training. This often leads to selecting a model which is suboptimal (although still provides acceptable results nonetheless). A future research direction would be on how the algorithm can be improved so that such noise can be mitigated.

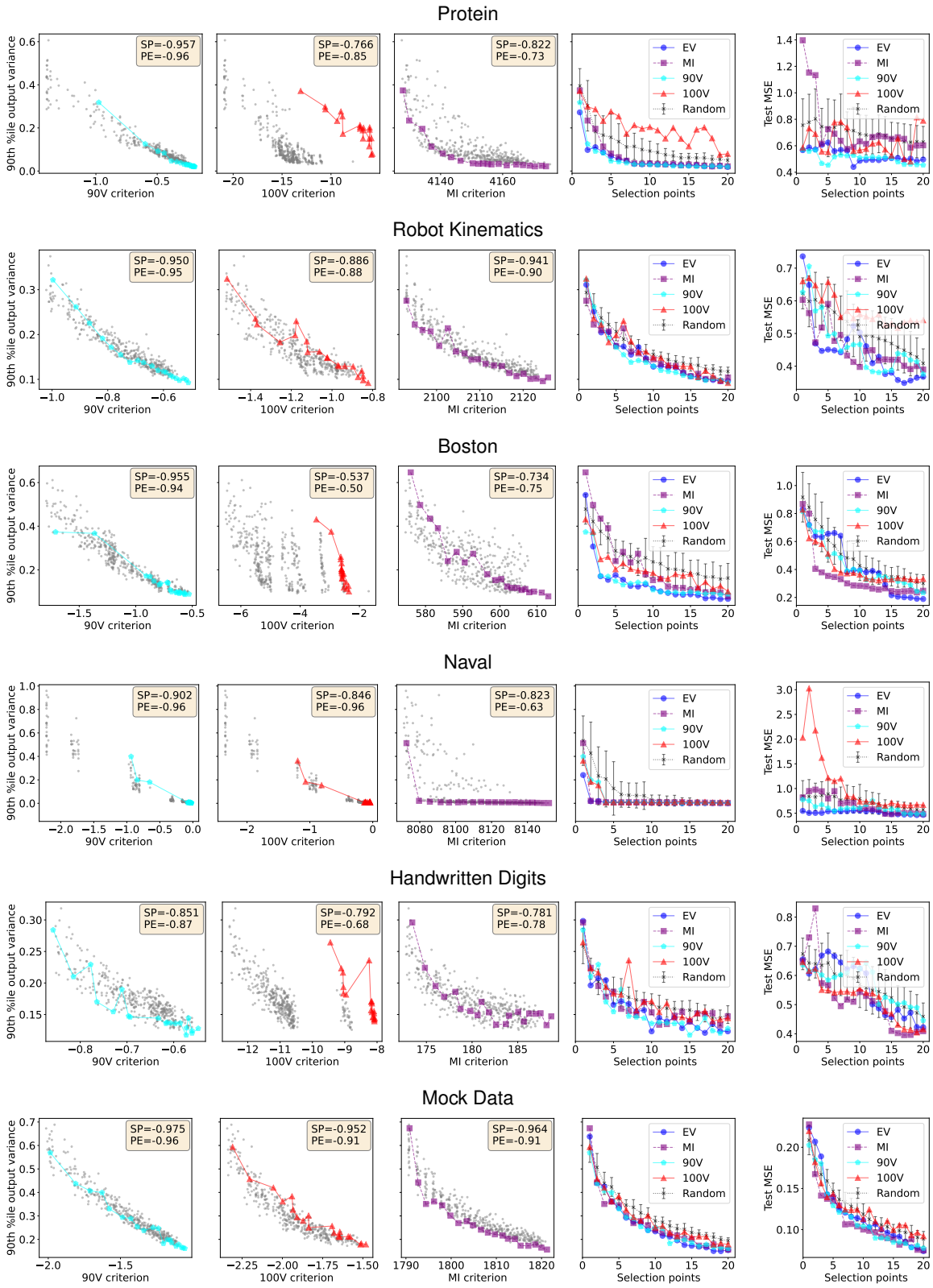

Figure 14. Relationship between the different criteria value, the 90th percentile predictive variance, and the test MSE. Each grey point represents one instance of a randomly-selected active set. The coloured line represents the subset greedily selected to optimize the corresponding criteria. Each row represents trials conducted on one dataset. SP and PR represents the Spearman Rank correlation and Pearson correlation coefficients respectively. The metrics reported on the y-axis are described further in App. G.3.

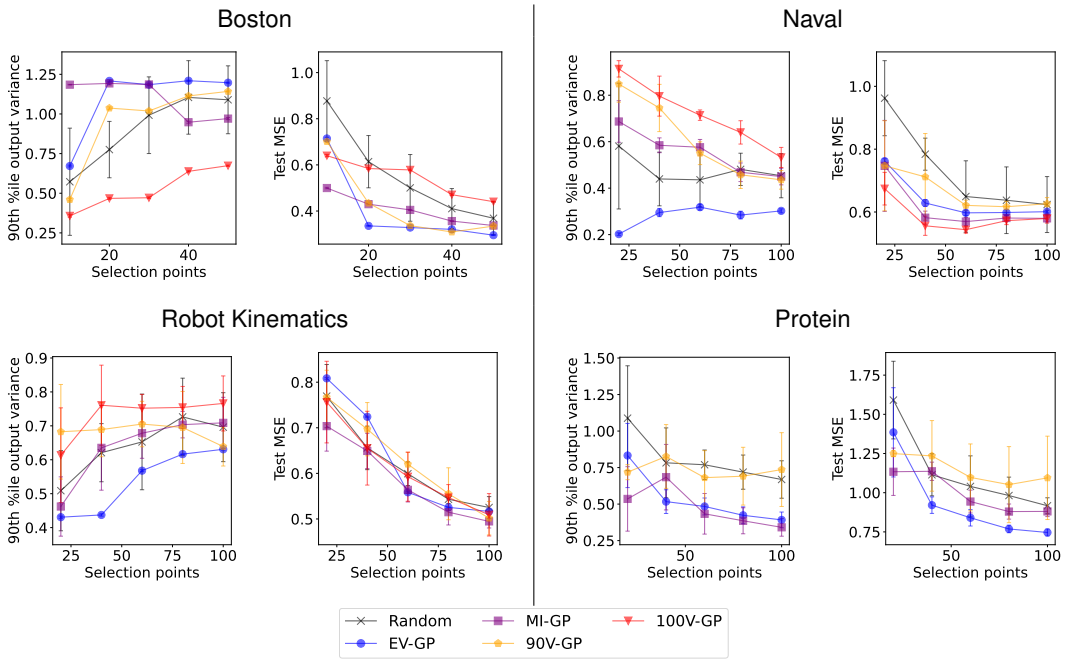

Figure 15. Active learning on regression datasets with different criteria based on σ^2_{NTKGP}. The information shown is presented in the same manner as Fig. 4 (see App. H.1).

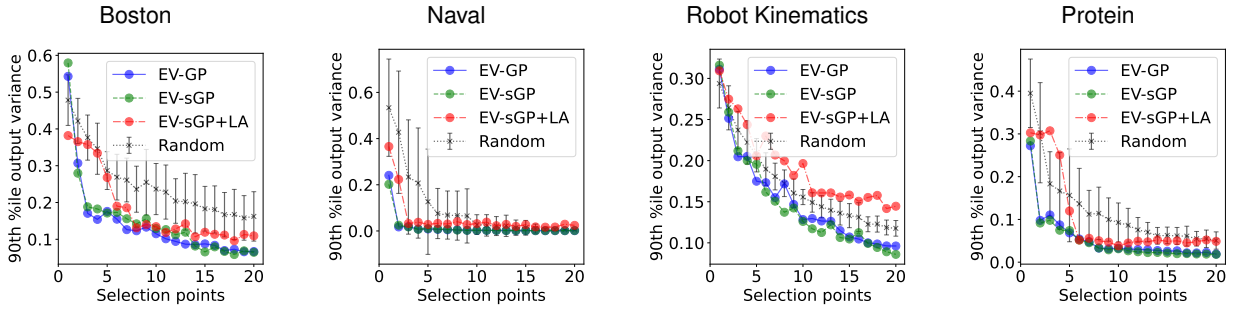

Figure 16. Active learning on regression problems with different approximations of σ_{NTKGP}. The graph shows the output variance of the test dataset when trained on the selected active set of various sizes, according to how it was approximated. The information presented is similar to that shown in the graphs on the right column of Fig. 3 (see App. H.1).

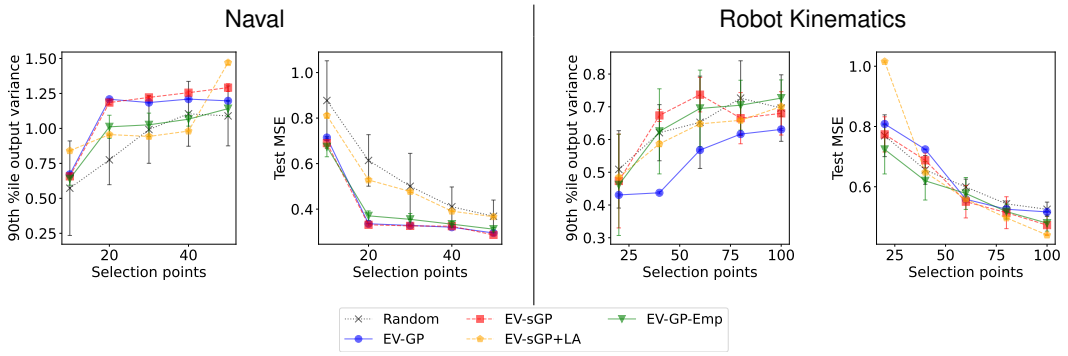

Figure 17. Active learning on regression datasets with different approximations of σ_{NTKGP} and with the empirical NTK. The information shown is presented in the same manner as Fig. 4 (see App. H.1).

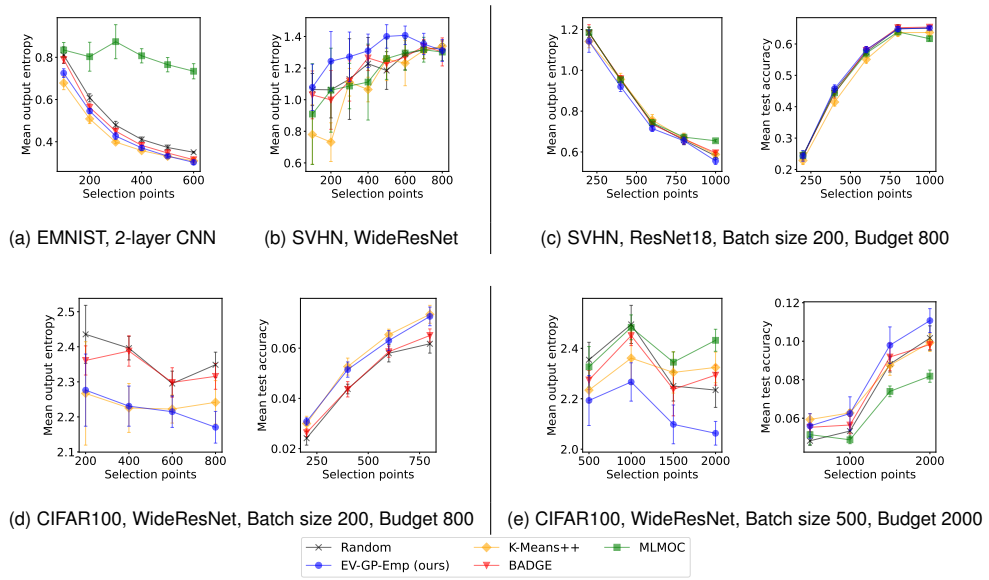

Figure 18. (a-b) Achieved output entropy for active learning on corresponding classification experiments conducted for Fig. 5(c-d). (c) Additional results on classification using ResNet18 on SVHN dataset. (d-e) Additional results on classification using WideResNet on CIFAR100 with different batch size and query budgets.

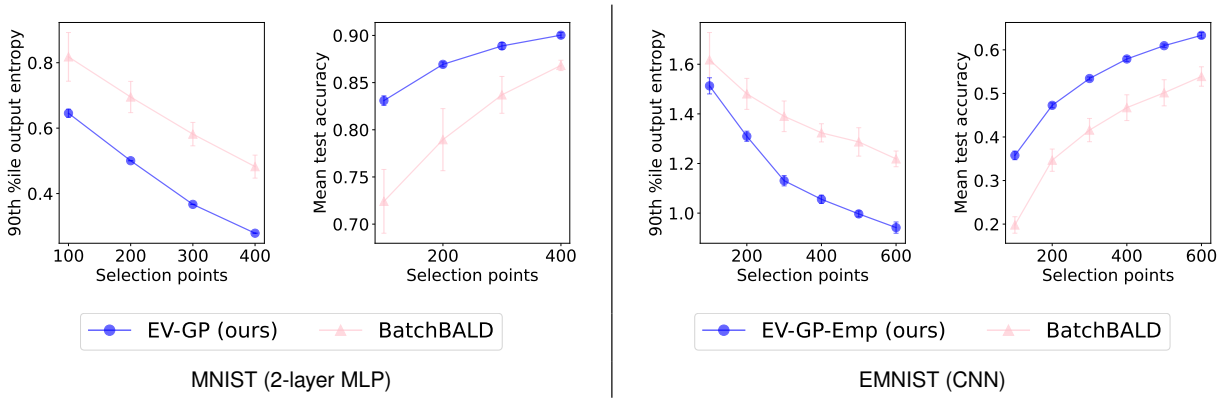

Figure 19. Comparison between EV-GP and BATCHBALD.

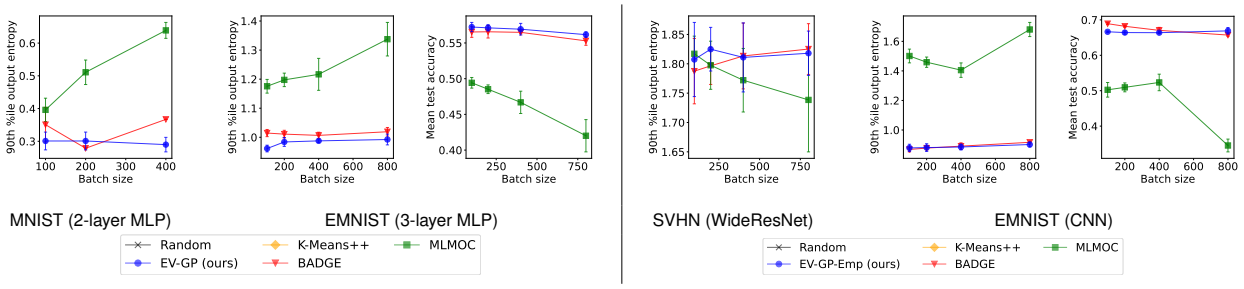

Figure 20. Results on classification with varying batch sizes.

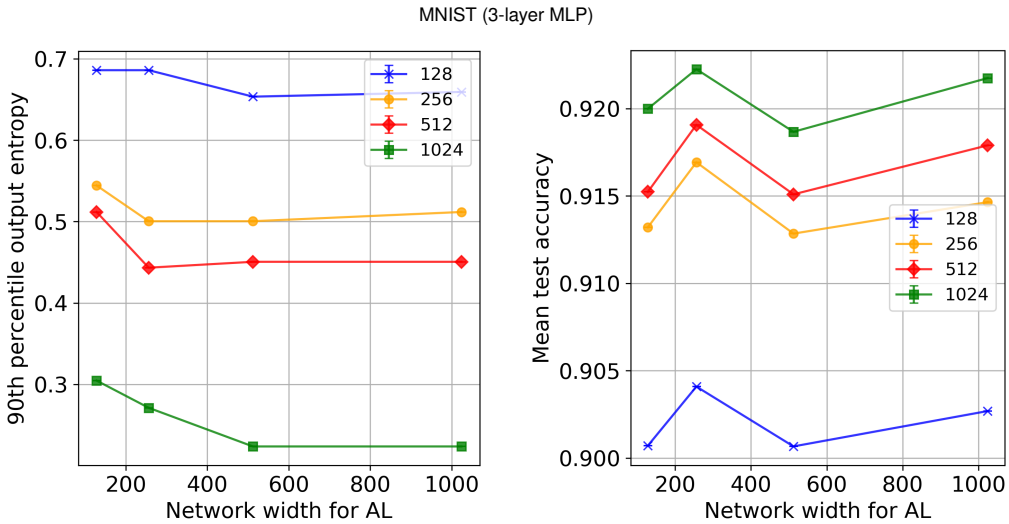

Figure 21. Performances of our EV-GP-EMP with varying NN widths for data selection (x-axis) and NN training (different lines).

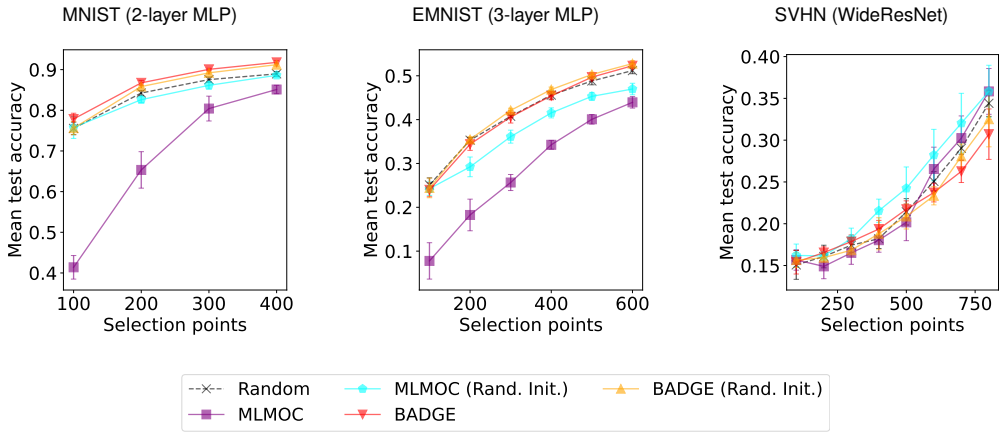

Figure 22. Results on classification with BADGE and MLMOC when random initial data is given. In the examples with (Rand. Init.) quantifier, the algorithm randomizes the first batch of data instead of using their algorithm to first select the points.

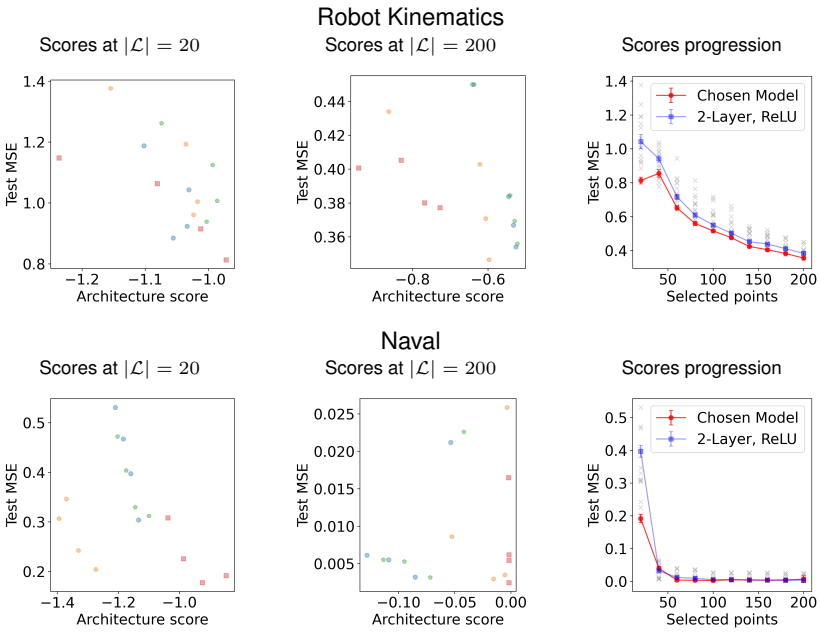

Figure 23. Further results related to EV-GP+MS from ablation experiments conducted for Fig. 7. The graphs presents how EV-GP+MS selects a model during one trial of active learning. *Left and center:* the plots between α_M and the obtained test MSE for each model architecture at different iterations of the active learning process. Points with the same shapes are models with the same activation functions but varying depth. *Right:* visualization of which models are selected by EV-GP+MS in each round. The gray crosses represent the resulting training when using a particular architecture from $\mathcal{M}$. The red line represents the choice from EV-GP+MS, and the blue line represents the training if we stuck with the 2-layer MLP with ReLU activation throughout.

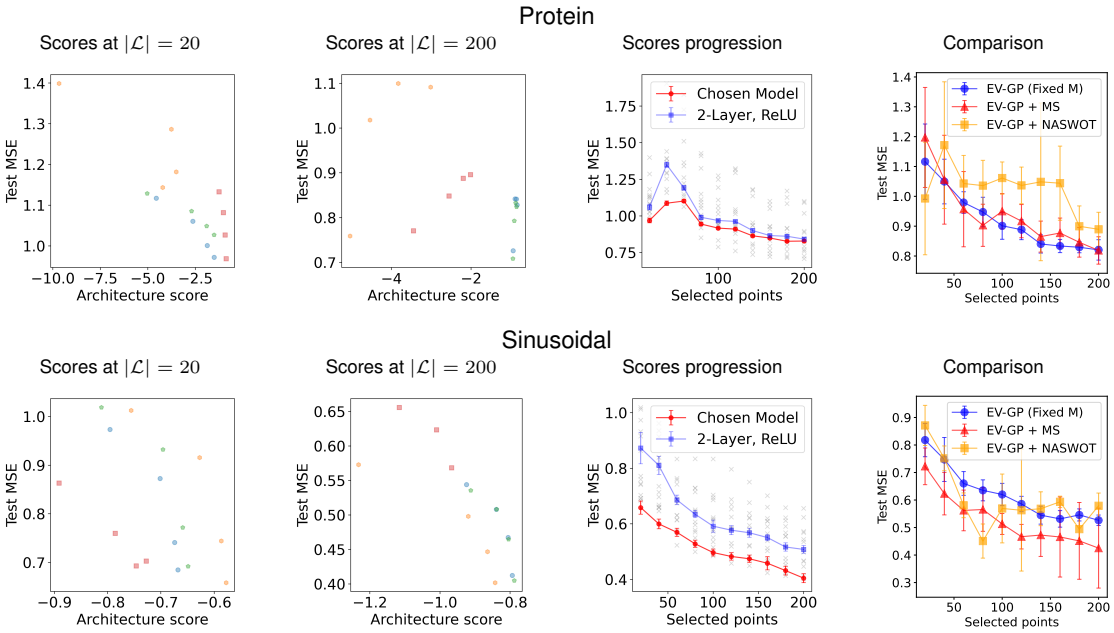

Figure 24. Additional experiments for EV-GP+MS conducted on other datasets. The data presented is presented in the same manner as Figs. 7 and 23.