Near-Optimal Task Selection for Meta-Learning with Mutual Information and Online Variational Bayesian Unlearning

Yizhou Chen
National University of Singapore

Shizhuo Zhang
Nanyang Technological University

Bryan Kian Hsiang Low
National University of Singapore

Abstract

This paper addresses the problem of active task selection which involves selecting the most informative tasks for meta-learning. We propose a novel active task selection criterion based on the mutual information between latent task vectors. Unfortunately, such a criterion scales poorly in the number of candidate tasks when optimized. To resolve this issue, we exploit the submodularity property of our new criterion for devising the first active task selection algorithm for meta-learning with a near-optimal performance guarantee. To further improve our efficiency, we propose an online variant of the Stein variational gradient descent to perform fast belief updates of the meta-parameters via maintaining a set of forward (and backward) particles when learning (or unlearning) from each selected task. We empirically demonstrate the performance of our proposed algorithm on real-world datasets.

1 Introduction

Meta-learning (also known as few-shot learning) exploits the experience from previous tasks to form a model (represented by meta-parameters) that can rapidly adapt to a new task with its few-shot data. Though meta-training requires only a few data points from each task, these tasks have to be representative of the distribution of meta-test tasks (Luna and Leonetti, 2020) in order to attain strong generalization performance for any meta-test task. To achieve this, one can naively consider acquiring (the data associated with) a massive number of tasks like that in various benchmark datasets, which in practice is prohibitively costly in time and money (e.g., due to manual labeling or data anonymization effort). Given a limited budget of k tasks to be acquired, one can simply select them randomly, but this often leads to a sub-optimal meta-test performance (Liu *et al.*, 2020). This motivates the problem of *active task selection* which involves selecting a subset of k most informative tasks from a finite yet large collection of candidate tasks for meta-training. These informative tasks are expected to be most representative of the distribution of meta-test tasks among any other k tasks and hence allow meta-training to generalize best to any meta-test task.[1]

To address the active task selection problem, we start by choosing a probabilistic meta-learning framework for grounding our active task selection criterion based on information theory. From the existing works on probabilistic meta-learning (Chen *et al.*, 2021; Finn *et al.*, 2018; Nguyen *et al.*, 2021a; Rusu *et al.*, 2019; Yoon *et al.*, 2018), we choose the *implicit process-based meta-learning* (IPML) framework (Chen *et al.*, 2021) (Sec. 2) that explicitly represents each task as a continuous latent vector and models its probabilistic belief within the highly expressive *implicit process* (IP) framework (Ma *et al.*, 2019). Representing each task as a latent task vector has a distinct advantage in that the dimension of the representation does not increase with the number of data points in a task, hence allowing active task selection criteria based on mutual information or entropy to be computed efficiently (Sec. 3.2). We then propose a novel active task selection criterion based on the *mutual information between latent task vectors* (MILT) to quantify the informativeness of any subset of tasks. Unfortunately, the MILT criterion scales poorly in the number of candidate tasks when optimized. To resolve this issue, we exploit the submodularity property of the MILT criterion for devising the first active task selection algorithm for meta-learning with a near-optimal performance guarantee (Sec. 3).

Our active task selection algorithm requires frequent

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

[1]Note that the motivation of active task selection is not to meta-learn faster in terms of wall-clock time.

belief updates of the meta-parameters, which can be computationally expensive. To further improve our efficiency, we design a forward-backward method based on our online variant of the *Stein variational gradient descent* (SVGD) (Liu and Wang, 2016) to perform fast belief updates such that a set of forward (and backward) particles is maintained and updated by learning (or unlearning) from each selected task (Sec. 4).

Our contributions are summarized as follows: (a) We propose the first active task selection algorithm for meta-learning with a near-optimal performance guarantee (Sec. 3); (b) We design a forward-backward method based on our online variant of SVGD to improve the efficiency of our algorithm (Sec. 4); (c) Empirical evaluation on several benchmark datasets demonstrate the state-of-the-art performance of our algorithm (Sec. 5).

2 Background and Notations

Among the existing probabilistic meta-learning frameworks (Chen *et al.*, 2021; Finn *et al.*, 2018; Nguyen *et al.*, 2021a; Yoon *et al.*, 2018), we adopt that of Chen *et al.* (2021) which explicit represents each task as a continuous latent vector and models its probabilistic belief. Following that of Chen *et al.* (2021), the inputs (outputs) for all candidate tasks are assumed to belong to the same input (output) space. Consider meta-learning on probabilistic regression tasks:[2] Each candidate task is assumed to be generated from a task distribution and associated with a dataset $(X, \mathcal{Y} = \mathbf{y})$ where X is a set of known/fixed input vectors,[3] $\mathcal{Y} \triangleq (y_{\mathbf{x}})^\top_{\mathbf{x}\in X}$ denotes a vector of the corresponding noisy outputs (random variables): $y_{\mathbf{x}} \triangleq f(\mathbf{x}) + \epsilon(\mathbf{x})$, which are outputs of an unknown function f corrupted by an i.i.d. Gaussian noise $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma^2)$ with variance σ^2, and the vector $\mathbf{y}$ denotes a realization of $\mathcal{Y}$. Let f be distributed by an *implicit process* (IP) (Ma *et al.*, 2019), as follows:

Definition 1 (**Implicit process (IP)**). *Let the collection of random variables $f(\cdot)$ denote an IP defined by meta-parameters (random variables) Θ: Every finite collection $\{f(\mathbf{x})\}_{\mathbf{x}\in X}$ has a joint prior belief $p(\mathbf{f} \triangleq (f(\mathbf{x}))^\top_{\mathbf{x}\in X})$ implicitly defined as:*

$$\mathcal{Z} \sim P(\mathcal{Z} = \mathbf{z}) = p(\mathbf{z}), \quad f(\mathbf{x}) \triangleq g_\Theta(\mathbf{x}, \mathcal{Z}) \qquad (1)$$

for all $\mathbf{x} \in X$ where every latent task vector $\mathcal{Z} = \mathbf{z}$ (representing a task) is drawn from the prior belief $p(\mathbf{z}) \triangleq \mathcal{N}(0, \mathbf{I})$, and generator g_Θ can be an arbitrary model (in our work here, a deep neural network (DNN)) parameterized by meta-parameters Θ.

[2]We defer the discussion of meta-learning on probabilistic classification tasks using the robust-max likelihood (Hernández-Lobato *et al.*, 2011) to Appendix A.

[3]From now on, for terms conditioned on the known/fixed inputs, we omit the inputs to ease notations.

Let the meta-parameters (random variables) Θ follow a prior belief $P(\Theta = \theta) = p(\theta)$. The goal of meta-learning is to infer the posterior belief $P(\Theta|\mathcal{Y} = \mathbf{y})$ of meta-parameters Θ. In contrast to the work of Chen *et al.* (2021) which adopts a point estimate of Θ, we consider a Bayesian treatment of the meta-parameters, which is empirically shown to improve performance (Sec. 5). Using $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2\mathbf{I})$ and the IP prior belief $p(\mathbf{f})$ from Def. 1, the marginal likelihood $P(\mathcal{Y} = \mathbf{y})$ can be derived by marginalizing out $\mathbf{f}$. Following that of Chen *et al.* (2021), the coupling of $\mathcal{Z}$ with the IP model is by masking the last DNN layer's parameters (i.e., point-wise product) with $\mathcal{Z}$ during forward propagation (Fig. 1c).

Notations. Let T denote a finite collection of candidate tasks. For each candidate task $t \in T$, we consider a split of its dataset $(X_t, \mathcal{Y}_t = \mathbf{y}_t)$ into a small *sample* dataset[4] $(X^s_t, \mathcal{Y}^s_t = \mathbf{y}^s_t)$ known *a priori* and a large *remaining* dataset $(X^r_t, \mathcal{Y}^r_t = \mathbf{y}^r_t)$ to be acquired/observed *only after* this task is selected by an active task selection algorithm. So, $X_t = X^s_t \cup X^r_t$ and $X^s_t \cap X^r_t = \emptyset$. Such a split is similar to the support-query split of a meta-training task (Finn *et al.*, 2018; Gordon *et al.*, 2019; Ravi and Beatson, 2018; Yoon *et al.*, 2018), albeit serving a different aim of active task selection here. For a subset $A \subseteq T$ of tasks, let $\mathcal{Y}^r_A \triangleq (\mathcal{Y}^r_t)_{t\in A}$ and $\mathbf{y}^r_A \triangleq (\mathbf{y}^r_t)_{t\in A}$ denote a realization of $\mathcal{Y}^r_A$. Similarly, let $\mathcal{Z}_A \triangleq (\mathcal{Z}_t)_{t\in A}$ concatenate the latent task vectors representing the tasks in A. Let $A \cup t$ denote the union of A and $\{t\}$.

3 Active Task Selection

Problem Definition. Given a finite collection T of candidate tasks with small, *a priori* known sample datasets $(X^s_t, \mathbf{y}^s_t)$ for all $t \in T$ (Sec. 2),[5] the problem of *active task selection* is about selecting a subset $A \subseteq T$ of *most informative* tasks subject to a budget of $|A| = k$ tasks. After selecting A, the remaining datasets $(X^r_t, \mathbf{y}^r_t)$ for all tasks $t \in A$ are acquired/observed for meta-training.

Note that in this work, we assume the candidate tasks are fixed prior to our selection. So, we are not allowed to generate our own tasks by selecting classes for meta-learning of classification tasks, like in (Liu *et al.*, 2020). Also, we do not assume the availability of any contextual information (e.g., hyperparameters generating task data) on the tasks, like in (Kaddour *et al.*, 2020). Our only assumption is the availability of known sample datasets for each task, which is real-

[4]For N-way K-shot classification problems, the sample dataset has K data points per class and N classes.

[5]From now on, for terms conditioned on the *a priori* known sample datasets, we omit them to ease notations.

istic and easy to achieve in practice. An example of our problem setting is when a meta-learner wants to purchase data from decentralized data marketplaces. These decentralized data marketplaces enable a secure data exchange between the participants. Powered by blockchain, they can maintain the anonymity of their participants which is why they are often used for the trading of personal data. They usually offer a period of free subscription such that the sample datasets can be easily acquired. In reality, such a decentralized data marketplace can be found in established data sharing platforms like Streamr Marketplace (`https://streamr.network/marketplace`) which offers data like real-time finance market data, and HIT Foundation (Eberhard and Paul, 2019) which offers healthcare data like radiology image data.

Meta-Learning Algorithms used after Active Task Selection. Given the selected A, in principle, we are free to use any meta-learning algorithms to obtain a final model, including probabilistic (Finn *et al.*, 2018; Nguyen *et al.*, 2021a; Yoon *et al.*, 2018) or non-probabilistic (Finn *et al.*, 2017) methods. We ground our active task selection criterion on IP. Probabilistic meta-learning of IP can be defined as the inference of meta-parameters Θ and the work of Chen *et al.* (2021) has used an *expectation maximization* (EM) algorithm to perform meta-training such that the E step carries out the IP inference of $\mathcal{Z}$ while the M step optimizes Θ. Since our focus is on active task selection instead of deriving a meta-learning algorithm, a detailed discussion of the meta-learning algorithms used in this paper can be found in Appendix C. Fig. 1 shows the graphical models of the IP for task selection as well as meta-learning.

3.1 MILT Criterion

To quantify the informativeness of a subset A of tasks, our proposed active task selection criterion measures its reduction in uncertainty/entropy of the latent task vectors $\mathcal{Z}_{T\setminus A}$ representing the other candidate tasks in $T\setminus A$ or, equivalently, its information gain $I(\mathcal{Z}_A;\mathcal{Z}_{T\setminus A})$ on them. We use the prior entropy $H(\mathcal{Z}_{T\setminus A})$ and posterior entropy $H(\mathcal{Z}_{T\setminus A}|\mathcal{Z}_A)$ to represent the uncertainty of $\mathcal{Z}_{T\setminus A}$ before and after conditioning on $\mathcal{Z}_A$. Then,

$$I(\mathcal{Z}_A;\mathcal{Z}_{T\setminus A}) \triangleq H(\mathcal{Z}_{T\setminus A}) - H(\mathcal{Z}_{T\setminus A}|\mathcal{Z}_A)\ , \qquad (2)$$

which we call the *mutual information between latent task vectors* (MILT) criterion. For simplicity, we define the set function $\mathrm{MILT}(A) \triangleq I(\mathcal{Z}_A;\mathcal{Z}_{T\setminus A})$. The k most informative candidate tasks thus correspond to the subset $A^\star \subseteq T$ with the largest information gain $\mathrm{MILT}(A^\star)$ on $\mathcal{Z}_{T\setminus A^\star}$:

$$A^\star \triangleq \arg\max_{A\subseteq T,|A|=k} \mathrm{MILT}(A)\ . \qquad (3)$$

Unfortunately, the MILT criterion (2) scales poorly in the number $|T|$ of candidate tasks when optimized in (3). In fact, solving (3) has been shown to be NP-hard even when $\mathcal{Z}_T$ follows a tractable multivariate Gaussian distribution (Ko *et al.*, 1995). Inspired by the work of Krause *et al.* (2008), we will now present a polynomial-time greedy algorithm that can exploit the submodularity of MILT to guarantee a $(1-1/e)$-factor approximation of that achieved by $A^\star$:

Algorithm I (**Near-optimal active task selection (informal)**). *Start with an empty set $A_0=\emptyset$ of tasks. In each round $i=1,\ldots,k$, greedily select the next task:*

$$t_i^\star \triangleq \arg\max_t \mathrm{MILT}(A_{i-1}\cup t) - \mathrm{MILT}(A_{i-1}) \qquad (4)$$

and update the set $A_i = A_{i-1}\cup t_i^\star = \{t_1^\star,\ldots,t_i^\star\}$ of selected tasks.

Theorem 1 (**Near-optimal guarantee**). *Algorithm I is guaranteed to select a set A of k tasks s.t.*

$$\mathrm{MILT}(A) \geq (1-1/e)(\mathrm{OPT} - C_0) \qquad (5)$$

where $\mathrm{OPT} \triangleq \mathrm{MILT}(A^\star)$ and the constant $C_0 \triangleq H(\Theta)$ is the prior entropy of meta-parameters Θ.

Its proof is in Appendix D.1. Note that for a monotonic submodular set function, a greedy algorithm can be designed to guarantee a $(1-1/e)$-factor approximation of OPT (Krause and Golovin, 2014). Though the $\mathrm{MILT}(\cdot)$ function is submodular, it is not strictly monotonic. Nevertheless, we can show that $\mathrm{MILT}(\cdot)$ is approximately monotonic up to a constant error of C_0, which suffices for the proof of Theorem 1. This approximate monotonicity holds due to $\mathcal{Z}_t, \mathcal{Z}_{t'}, \forall t\neq t'$ being mutually independent given meta-parameters Θ.

3.2 Advantages of MILT over other Active Task Selection Criteria

For our active task selection problem, other criteria can be considered. For example, A can be selected to maximize the *mutual information between remaining datasets of tasks* (MIRD) criterion $I(\mathcal{Y}_A^r;\mathcal{Y}_{T\setminus A}^r) \triangleq H(\mathcal{Y}_{T\setminus A}^r) - H(\mathcal{Y}_{T\setminus A}^r|\mathcal{Y}_A^r)$. In fact, we can also prove a similar near-optimal performance guarantee for MIRD, as shown in Appendix D.2. However, any criterion involving probabilities conditioned on $\mathcal{Y}_A^r$ tends to be computationally challenging to evaluate: Suppose that each task has a potentially large remaining dataset of size R. Since the dimension of $\mathcal{Y}_A^r$ is proportional to $R|A|$, it becomes computationally challenging to compute an accurate Monte Carlo estimation of MIRD. So, it is not computationally feasible to evaluate the MIRD criterion when R is large. In contrast, MILT does not suffer from this curse of dimensionality since

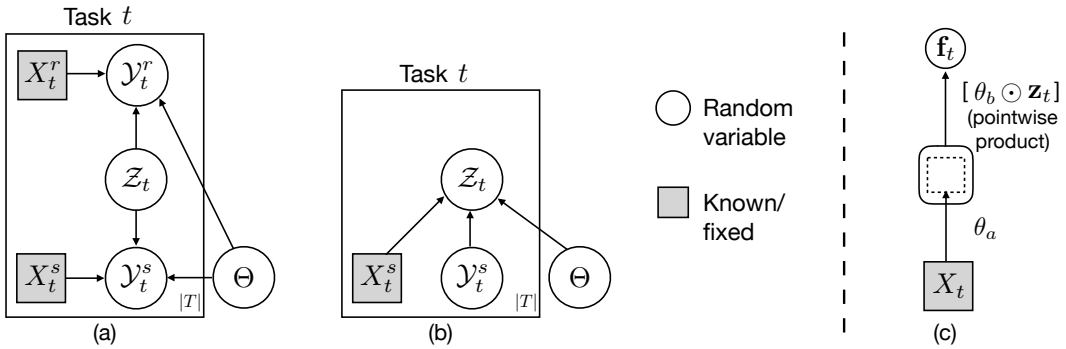

Figure 1: Graphical model of implicit process for (a) meta-learning and (b) task selection. We omit the random variable $f(\cdot)$ to simplify illustration since $f(\cdot) \to \mathcal{Y}$ is simply the addition of i.i.d. Gaussian noises. (c) DNN generator g_θ where $\theta \triangleq (\theta_a, \theta_b)$ and θ_a can be convolutions to obtain high-level representations of the input vector, while θ_b is the last DNN layer's parameters.

the dimension of a latent task vector $\mathcal{Z}$ in (1) does not increase with R.[6]

As another example, A can be selected to maximize the *entropy of latent task vectors* (ELT) criterion $H(\mathcal{Z}_A)$. However, directly maximizing ELT is NP-hard even when $\mathcal{Z}_T$ follows a tractable multivariate Gaussian distribution (Ko *et al.*, 1995). So, we usually resort to greedily selecting $\arg\max_t H(\mathcal{Z}_t|\mathcal{Z}_{A_{i-1}})$ for $i = 1, \ldots, k$, which has a comparable computational cost as Algo. I. It is well-known that although the entropy criterion is submodular, it may not be monotonic (Krause and Golovin, 2014). Hence, such a greedy algorithm based on ELT does not enjoy the near-optimal performance guarantee and often performs sub-optimally (Krause *et al.*, 2008).

The work of Chen *et al.* (2021) has proposed to greedily select the next task with the maximum variance of $\mathcal{Z}_t$: $t_i^\star = \arg\max_t \mathrm{Var}(p(\mathbf{z}_t|\theta, \mathbf{y}_t^s))$ for $i = 1, \ldots, k$. However, doing so neglects the dependence of $\mathcal{Z}_t$ on $\mathcal{Z}_{A_{i-1}}$; it can be observed from the Fig. 1b that they are dependent when $\mathcal{Y}_t^s$ and $\mathcal{Y}_{A_{i-1}}^s$ are both observed. To correctly account for such a dependence, the following variance criterion should be considered instead: $t_i^\star = \arg\max_t \mathrm{Var}(\mathcal{Z}_t|\mathcal{Z}_{A_{i-1}})$ for $i = 1, \ldots, k$, which has a similar computational cost as the greedy algorithm based on ELT, but does not have any performance guarantee. A comparison of the above-discussed active criteria are summarized in Table 1.

4 Efficient Evaluation of (4)

Let $\overline{A} \triangleq T \setminus A$. From (4),

$$\begin{aligned} & \mathrm{MILT}(A_{i-1} \cup t) - \mathrm{MILT}(A_{i-1}) \\ = \; & H(\mathcal{Z}_t|\mathcal{Z}_{A_{i-1}}) - H(\mathcal{Z}_t|\mathcal{Z}_{\overline{A_{i-1} \cup t}}) \; . \end{aligned} \tag{6}$$

In (6), the posterior entropy $H(\mathcal{Z}_t|\mathcal{Z}_A) = -\mathbb{E}_{\mathbf{z}_A \sim p(\mathbf{z}_A)} \left[\int_{\mathbf{z}_t} p(\mathbf{z}_t|\mathbf{z}_A) \log p(\mathbf{z}_t|\mathbf{z}_A) \, \mathrm{d}\mathbf{z}_t\right]$ can be approximated by Monte Carlo sampling from $p(\mathbf{z}_A)$ such that for each sample of $\mathbf{z}_A$, we compute

$$p(\mathbf{z}_t|\mathbf{z}_A) = \int_\theta \underbrace{p(\theta|\mathbf{z}_A, \mathbf{y}_A^s)}_{\text{particles (Sec. 4.1)}} \underbrace{p(\mathbf{z}_t|\theta, \mathbf{y}_t^s)}_{\text{Gaussian (Sec. 4.2)}} \mathrm{d}\theta \; . \tag{7}$$

From (7), the selection of A affects that of the next task t through the common meta-parameters Θ in the IP model, as shown in Fig. 1.

4.1 Variational Inference (VI) with Particle Representation of Θ

In this subsection, we will describe the procedure to obtain the approximation of $p(\theta|\mathbf{z}_A, \mathbf{y}_A^s)$. We use a particle representation of Θ and a stochastic VI method for particles called the *Stein variational gradient descent* (SVGD) (Liu and Wang, 2016) to compute the posterior belief of Θ. Applied in a previous Bayesian meta-learning framework (Yoon *et al.*, 2018), SVGD combines the strengths of MCMC and variational inference and enjoys a similar time efficiency as stochastic gradient descent. SVGD represents the belief of Θ with a set $\{\theta^m\}_{m=1}^M$ of M particles.

Though $p(\theta|\mathbf{z}_A, \mathbf{y}_A^s)$ in (7) cannot be evaluated in closed form, it can be computed via SVGD: Its gradient $\nabla_\theta \log p(\theta|\mathbf{z}_A, \mathbf{y}_A^s) = \nabla_\theta \log[p(\mathbf{y}_A^s|\theta, \mathbf{z}_A)p(\theta)]$ is available given our neural network implementation of IP: $p(\mathbf{y}_A^s|\theta, \mathbf{z}_A) = \prod_{t \in A} p(\mathbf{y}_t^s|\mathbf{f}_t^s = (g_\theta(\mathbf{x}, \mathbf{z}_t))_{\mathbf{x} \in X_t^s}^\top)$ where g_θ is a neural network parameterized by θ (Def. 1). We perform SVGD on the observed tuples $\{\mathbf{z}_A, X_A^s, \mathbf{y}_A^s\}$ by first initializing each particle as a sample from $p(\theta)$[7] and then, in each SVGD iteration, updating each par-

[6] Such an advantage of latent task modeling motivates us to use the IP framework (Def. 1).

[7] Note that this is not strictly needed for SVGD to converge.

Table 1: Comparison of different active task selection criteria. The last column indicates the dimension of the sample space of the Monte Carlo sampling step when evaluating the active task selection criterion.

Criterion	Expression	Submodular	Approx. monotonic	Near-optimal	Sample space
MILT	$H(\mathcal{Z}_{T\setminus A}) - H(\mathcal{Z}_{T\setminus A}\|\mathcal{Z}_A)$	✓	✓	✓	$\mathcal{O}(k)$
MIRD	$H(\mathcal{Y}^r_{T\setminus A}) - H(\mathcal{Y}^r_{T\setminus A}\|\mathcal{Y}^r_A)$	✓	✓	✓	$\mathcal{O}(Rk)$
ELT	$H(\mathcal{Z}_A)$	✓	✗	✗	$\mathcal{O}(k)$
Variance	$\mathrm{Var}(\mathcal{Z}_t\|\mathcal{Z}_{A_{i-1}})$ for $i = 1,\ldots,k$	✗	✗	✗	$\mathcal{O}(k)$

ticle θ^m as

$$\theta^m \leftarrow \theta^m + \frac{\eta}{M}\sum_{\theta\in\{\theta^m\}_{m=1}^M}\big[k(\theta,\theta^m) \times\nabla_\theta \log p(\theta|\mathbf{z}_A,\mathbf{y}^s_A) + \nabla_\theta k(\theta,\theta^m)\big] \tag{8}$$

where η is the step size and $k(\cdot,\cdot)$ is a radial basis function kernel representing a repulsive force between particles to prevent them from collapsing. We denote the abstraction of the entire VI process (containing multiple SVGD iterations till convergence) to obtain $p(\theta|\mathbf{z}_A,\mathbf{y}^s_A)$ as $p(\theta|\mathbf{z}_A,\mathbf{y}^s_A) \leftarrow \mathrm{SVGD}_\Theta(p(\theta),\{\mathbf{z}_A,\mathbf{y}^s_A\})$ where the first input is the initialization of the particles and the second input is the observed tuples. By setting $M = 1$, we will recover the gradient descent method to compute the point estimate of Θ.

4.2 VI with Gaussian Approximation of $\mathcal{Z}$

To model $p(\mathbf{z}_t|\theta,\mathbf{y}^s_t)$ in (7), we use VI with a Gaussian approximation of the posterior belief which makes use of the gradient on $\mathcal{Z}$: For each particle θ^m, we perform VI to obtain the mean and variance parameterization[8] of a Gaussian distribution $q(\mathbf{z}_t|\theta^m)$ by maximizing the evidence lower bound $\mathrm{ELBO}_\mathcal{Z} \triangleq \mathbb{E}_{q(\mathbf{z}_t|\theta^m)}[\log p(\mathbf{y}^s_t|\mathbf{z}_t,\theta^m)] - \mathrm{KL}(q(\mathbf{z}_t|\theta^m)\|p(\mathbf{z}_t))$ where KL stands for Kullback–Leibler divergence. As a result, $p(\mathbf{z}_t|\mathbf{z}_A)$ will be a *mixture of Gaussians* (since $p(\theta|\mathbf{z}_A,\mathbf{y}^s_A)$ is a set of M particles), which allows us to arrive at an easy approximation of the posterior entropy and hence (6). We denote the whole VI process as: $p(\mathbf{z}_t|\mathbf{z}_A) \leftarrow \mathrm{VI}_\mathcal{Z}(p(\theta|\mathbf{z}_A,\mathbf{y}^s_A))$.

4.3 Forward-Backward Method for Efficiently Evaluating (6)

In this subsection, we will describe a forward-backward method based on online SVGD to perform fast belief updates of the meta-parameters, thus improving the efficiency in evaluating (6). When we proceed from round $(i-1)$ to i, we update the set $A_i = A_{i-1}\cup t^\star_i$ of selected tasks. Then, $p(\theta|\mathbf{z}_{A_i},\mathbf{y}^s_{A_i})$ can be correspondingly updated from $p(\theta|\mathbf{z}_{A_{i-1}},\mathbf{y}^s_{A_{i-1}})$ by learning from the (sample dataset of) newly added task $t^\star_i$ in an online manner:[9] $p(\theta|\mathbf{z}_{A_i},\mathbf{y}^s_{A_i}) \leftarrow \mathrm{SVGD}_\Theta(p(\theta|\mathbf{z}_{A_{i-1}},\mathbf{y}^s_{A_{i-1}}),\{\mathbf{z}_{t^\star_i},\mathbf{y}^s_{t^\star_i}\})$. To this end, we only maintain a single set of particles and update it in place. We refer to the particles in this set as *forward particles*. Note that a significant advantage here is that in practice, by learning only from the newly added task, we need much fewer SVGD iterations (around 5) to converge to $p(\theta|\mathbf{z}_{A_i},\mathbf{y}^s_{A_i})$ compared with naively learning $p(\theta|\mathbf{z}_{A_i},\mathbf{y}^s_{A_i})$ from scratch (i.e., from $p(\theta)$).

The same trick applies when we compute $H(\mathcal{Z}_t|\mathcal{Z}_{\overline{A_{i-1}\cup t}})$ in (6). Firstly, we obtain $p(\theta|\mathbf{z}_{\overline{A_i}},\mathbf{y}^s_{A_i})$ which can be updated from $p(\theta|\mathbf{z}_{\overline{A_{i-1}}},\mathbf{y}^s_{A_{i-1}})$ by *unlearning* from the task $t^\star_i$. As shown in (Nguyen *et al.*, 2020), unlearning in the variational Bayes setting can be cast exactly as a minimization of an *evidence upper bound* (EUBO)[10], which is also applicable to SVGD, as stated below:

Proposition 1 (Online SVGD for unlearning). *Suppose that $p(\theta)$ is an uninformative prior (e.g., $\nabla_\theta \log p(\theta) = 0$). With $\{\theta^m\}_{m=1}^M$ initially sampled from $p(\theta|\mathbf{z}_A,\mathbf{y}^s_A)$, the SVGD operation for obtaining $p(\theta|\mathbf{z}_{A\setminus t},\mathbf{y}^s_{A\setminus t})$ (i.e., unlearning from a task $t\in A$) is*

$$\theta^m \leftarrow \theta^m + \frac{\eta}{M}\sum_{\theta\in\{\theta^m\}_{m=1}^M}\big[-k(\theta,\theta^m) \times\nabla_\theta \log p(\theta|\mathbf{z}_t,\mathbf{y}^s_t) + \nabla_\theta k(\theta,\theta^m)\big]\ . \tag{9}$$

We denote such an unlearning process (containing multiple SVGD iterations till convergence) as $p(\theta|\mathbf{z}_{A\setminus t},\mathbf{y}^s_{A\setminus t}) \leftarrow SVGD^{-1}_\Theta(p(\theta|\mathbf{z}_A,\mathbf{y}^s_A),\{\mathbf{z}_t,\mathbf{y}^s_t\})$.

Note that SVGD for unlearning (9) differs from that for learning (8): The sign of the likelihood gradient is reversed due to unlearning, while the sign of the kernel gradient is not as it corresponds to a repulsive force term which remains the same in both Stein operators of learning and unlearning. The proof of Proposition 1 (Appendix D.4) is obtained by deriving the Stein operator (Liu and Wang, 2016) for the minimization of EUBO (i.e., unlearning). We make use of the unlearning variant of online SVGD by maintaining another set of *backward particles*, which is initialized

[8]The mean and variance are optimized using gradient descent via the reparametrization trick.

[9]A relevant proposition that formally describes this online SVGD is provided in Appendix D.3.

[10]See Appendix D.4 for more details.

Algorithm 1 Near-Optimal Active Task Selection based on MILT

Set $A = \emptyset$;
Initialize forward particles: $p(\theta|\mathbf{z}_A, \mathbf{y}_A^s) = p(\theta)$;
Initialize backward particles: $p\left(\theta|\mathbf{z}_{\overline{A}}, \mathbf{y}_{\overline{A}}^s\right) = p(\theta|\mathbf{z}_T, \mathbf{y}_T^s) \leftarrow \text{SVGD}_\Theta\left(p(\theta), \{\mathbf{z}_T, \mathbf{y}_T^s\}\right)$;
while $|A| < k$ **do**
  Sample $\mathbf{z}_A \sim p(\mathbf{z}_A)$, $\mathbf{z}_{\overline{A}} \sim p(\mathbf{z}_{\overline{A}})$;
  for $t \in T \setminus A$ **do**
    Compute with forward particles: $p(\mathbf{z}_t|\mathbf{z}_A) \leftarrow \text{VI}_\mathcal{Z}(p(\theta|\mathbf{z}_A, \mathbf{y}_A^s))$;
    With backward particles: $p(\mathbf{z}_t|\mathbf{z}_{\overline{A\cup t}}) \leftarrow \text{VI}_\mathcal{Z}\left(\text{SVGD}_\Theta^{-1}\left(p\left(\theta|\mathbf{z}_{\overline{A}}, \mathbf{y}_{\overline{A}}^s\right), \{\mathbf{z}_t, \mathbf{y}_t^s\}\right)\right)$;
    Estimate $H(\mathcal{Z}_t|\mathcal{Z}_A) - H(\mathcal{Z}_t|\mathcal{Z}_{\overline{A\cup t}})$;
  end for
  Select $t^\star = \arg\max_t H(\mathcal{Z}_t|\mathcal{Z}_A) - H(\mathcal{Z}_t|\mathcal{Z}_{\overline{A\cup t}})$;
  Update forward particles: $p(\theta|\mathbf{z}_{A\cup t^\star}, \mathbf{y}_{A\cup t^\star}^s) \leftarrow \text{SVGD}_\Theta\left(p(\theta|\mathbf{z}_A, \mathbf{y}_A^s), \{\mathbf{z}_{t^\star}, \mathbf{y}_{t^\star}^s\}\right)$;
  Update backward particles: $p\left(\theta|\mathbf{z}_{\overline{A\cup t^\star}}, \mathbf{y}_{\overline{A\cup t^\star}}^s\right) \leftarrow \text{SVGD}_\Theta^{-1}\left(p\left(\theta|\mathbf{z}_{\overline{A}}, \mathbf{y}_{\overline{A}}^s\right), \{\mathbf{z}_{t^\star}, \mathbf{y}_{t^\star}^s\}\right)$;
  Update $A = A \cup t^\star$;
end while
return A

as $p(\theta|\mathbf{z}_T, \mathbf{y}_T^s)$. We then update it in place in every round of task selection through unlearning from task $t_i^\star$. Note that unlearning from a single task needs much fewer SVGD iterations (around 5) to converge to $p(\theta|\mathbf{z}_{\overline{A_i}}, \mathbf{y}_{\overline{A_i}}^s)$ compared with learning it from scratch (i.e., from $p(\theta)$). Unlearning can also be applied to obtain $p(\theta|\mathbf{z}_{\overline{A_i\cup t}}, \mathbf{y}_{\overline{A_i\cup t}}^s)$ (i.e., for $H(\mathcal{Z}_t|\mathcal{Z}_{\overline{A_{i-1}\cup t}})$ in (6)) such that $p(\theta|\mathbf{z}_{\overline{A_i\cup t}}, \mathbf{y}_{\overline{A_i\cup t}}^s)$ is obtained by unlearning $p(\theta|\mathbf{z}_{\overline{A_i}}, \mathbf{y}_{\overline{A_i}}^s)$ from task t.

Time complexity. Fig. 2 shows a computational graph of evaluating (6). Algo. 1 describes a detailed variant of our near-optimal active task selection algorithm that utilizes the forward-backward method. We show in Appendix B that its computational cost scales linearly in $|T|$. A detailed variant of the algorithm that uses the *naive* method (without the forward-backward method) is included in Appendix B where we show that its computational cost scales quadratically in $|T|$.

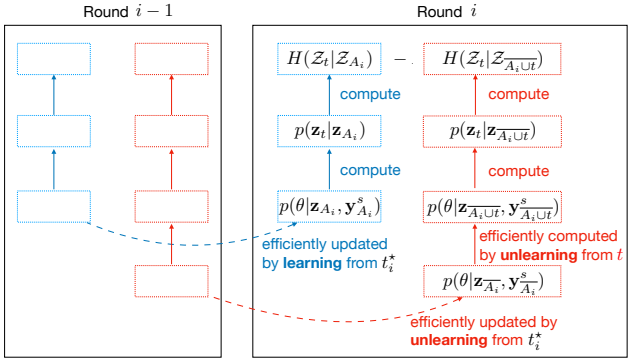

Figure 2: Computational graph of evaluating (6) with the forward-backward method in Sec. 4.3. The blue segments are computed using forward particles while the red segments are computed using backward particles.

5 Experiments and Discussion

In this section, we will empirically compare the performance of three active task selection criteria listed in Table 1:[11] (a) greedy algorithm based on MILT (4), (b) greedy algorithm based on ELT (Sec. 3.2), (c) greedy algorithm based on an improved variance criterion over that of Chen *et al.* (2021) (Sec. 3.2), and also random task selection using three benchmark datasets.[12] We have also adapted the greedy class-pair based sampling (GCP) proposed in (Liu *et al.*, 2020), and the probabilistic active meta-learning algorithm (PAML) proposed in (Kaddour *et al.*, 2020) which have different problem settings, and compared with them in Appendix E.2.

Sinusoid regression. In this setting, the data of each regression task is sampled from a sinusoid wave where the amplitude and phase vary between tasks. Following the experimental setting of Finn *et al.* (2017), the amplitude varies within $[0.1, 5]$, the phase varies within $[0, \pi]$, and the input $\mathbf{x}$ is sampled uniformly from $[-5, 5]$. We perform experiments in both the 5-shot setting (i.e., $|X_t^s| = |X_t^r| = 5$) and the 10-shot setting (i.e., $|X_t^s| = |X_t^r| = 10$)[13]. We randomly sample 1000

[11] Note that the comparison does not include MIRD due to its potentially high computational cost (Sec. 3.2). We have also included a comparison with (Luna and Leonetti, 2020) in Appendix E.6.

[12] We use regression benchmark datasets as surrogate examples of real-time data like finance market data, and classification benchmark datasets as surrogate examples of healthcare radiology image data.

[13] We have investigated the effect of using a larger remain-

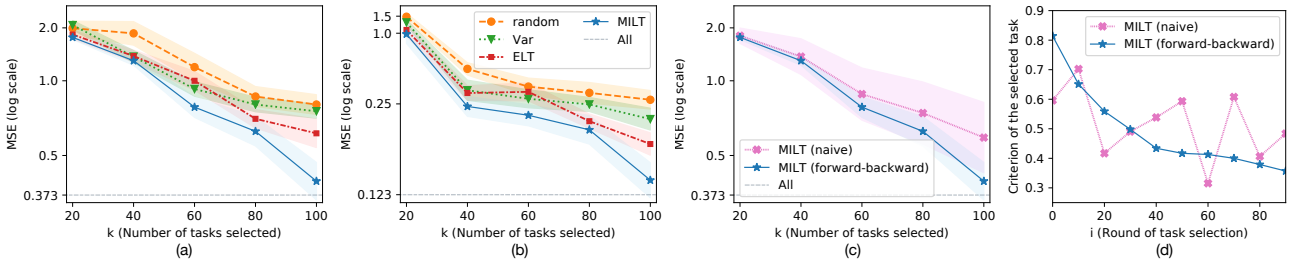

Figure 3: Meta-test *mean squared error* (MSE) and standard error over 5 runs vs. no. k of selected tasks on (a) 5-shot Sinusoid and (b) 10-shot Sinusoid. (c) Comparison of meta-test MSE between forward-backward method and naive method on 5-shot Sinusoid. *All* means meta-test MSE of the IP model trained on T (i.e., all 1000 candidate tasks). (d) plots the greedy criterion value $\text{MILT}(A_{i-1} \cup t_i^\star) - \text{MILT}(A_{i-1})$ (4) corresponding to selected task $t_i^\star$ vs. round i of selection.

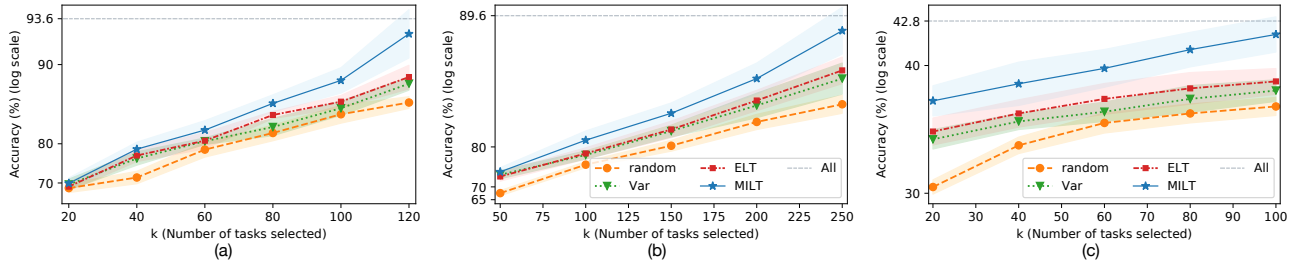

Figure 4: Meta-test accuracy (%) and standard error over 5 runs vs. no. k of selected tasks on (a) 1-shot 5-way Omniglot, (b) 1-shot 20-way Omniglot, and (c) 1-shot 5-way MiniImageNet. *All* means meta-test accuracy of the IP model trained on T (i.e., all 2000 candidate tasks).

tasks as T.

Omniglot classification. Omniglot (Lake *et al.*, 2011) is a benchmark few-shot image classification dataset consisting of 20 instances of 1623 characters from 50 different alphabets. Our experiment adopts the same task generation process as that in (Finn *et al.*, 2017) (i.e., downsampling to 28×28 and applying random rotations). To further accelerate computation, we select tasks in a *batch* manner: Each task consists of 32 sub-tasks such that each sub-task is a 1-shot 5-way (i.e., $|X_t^s| = |X_t^r| = 5$) or 1-shot 20-way (i.e., $|X_t^s| = |X_t^r| = 20$) classification.[14] We randomly generate 2000 tasks as T.

MiniImageNet classification. The MiniImageNet (Ravi and Larochelle, 2017) dataset involves 64 training classes, 12 validation classes, and 24 test classes of 84×84 RGB images. Our experiment adopts the same task generation process as that in (Finn *et al.*, 2017) (i.e., applying random rotations). Similarly, we select tasks in a *batch* manner: Each task consists of 32 sub-tasks such that each sub-task is a 1-shot 5-way (i.e., $|X_t^s| = |X_t^r| = 5$) classification.[14] We randomly generate 2000 tasks as T.

ing dataset in Appendix E.5.

[14]The active task selection criterion is a sum of the criterion over all sub-tasks.

The IP model (Sec. 2) is a fully-connected neural network with 2 hidden layers of size 40 with ReLU nonlinearities for Sinusoid, and a convolutional neural network with 4 modules of 3×3 convolutions and 64 filters, followed by batch normalization, ReLU nonlinearities, and strided convolutions for Omniglot or 2×2 max-pooling for MiniImageNet. We use $M = 5$ particles, and set the step size η as 0.05 for sinusoid and MiniImageNet and 0.5 for Omniglot.

5.1 Discussion of Baseline Comparisons

It can be observed from Figs. 3 and 4 that MILT outperforms all other baselines, which demonstrates the effectiveness of our proposed algorithm (Algo. 1). Random task selection performs the worst among all baselines, which is expected. ELT slightly outperforms the Variance baseline in nearly all cases, likely due to the entropy being able to better capture the uncertainty in $p(\mathbf{z}_t|\mathbf{z}_A)$ (7) which follows a mixture of Gaussians instead of a Gaussian.

Fig. 3 shows results of Sinusoid regression. Fig. 3b shows that actively selecting $k = 40$ tasks with MILT can already achieve a lower MSE of 0.241 than randomly selecting $k = 100$ tasks (MSE of 0.266). For both 5-shot (Fig. 3a) and 10-shot (Fig. 3b) settings, MILT achieves comparable performance to the IP model

trained with all 1000 candidate tasks in T when actively selecting only 10% from T (i.e., $k = 100$ tasks).

Fig. 4 shows results of Omniglot and MiniImageNet classifications. For both Omniglot and MiniImageNet, we have generated 2000 candidate tasks in T. Meta-training with all these candidate tasks achieves a meta-test accuracy of 93.6% for 1-shot 5-way Omniglot, 89.6% for 1-shot 20-way Omniglot, and 42.8% for 1-shot 5-way MiniImageNet (Fig. 4). Previous works (Chen *et al.*, 2021; Finn *et al.*, 2017, 2018; Yoon *et al.*, 2018) have generated 200000 tasks (batches) for meta-training and hence achieve higher meta-test accuracy on these cases. Nevertheless, our ablation study in Appendix E.3 shows that when selecting ($\sim$250) tasks, MILT does not need such a massive number ($\approx$ 200000) of candidate tasks to achieve competitive meta-test performance.

Fig. 4a shows that actively selecting only 120 tasks with MILT can achieve a meta-test accuracy of 91.9% for 1-shot 5-way Omniglot, which is within one standard error from that achieved by the IP model trained on all 2000 candidate tasks. Fig. 4b&c also show that for 1-shot 20-way Omniglot and 1-shot 5-way MiniImageNet, similar observations hold when selecting only 250 tasks with MILT. All these observations further demonstrate the effectiveness of our proposed algorithm (Algo. 1).

5.2 Ablation Study

Effect of using particles. Table 2 shows results of meta-test performance of MILT with varying number M of particles (Sec. 4.1) from 1 (point estimate) to 5. It can be observed that increasing M consistently yields better performance for both 5-shot and 10-shot Sinusoid regression, thus indicating that our Bayesian treatment of the meta-parameters can improve the meta-test performance.

Forward-backward method based on online SVGD. Here, we compare the performance of the forward-backward method (Algo. 1) with the naive method (Sec. 4.3). Table 3 presents a comparison of their runtime, which meets our theoretical analysis. Fig. 3c presents a comparison of their meta-test performance: It may be surprising to observe that the forward-backward method clearly outperforms the naive method. In theory, we should expect them to perform similarly since the forward-backward method only seems to improve the efficiency. However, in practice, the forward-backward method performs better because it does not introduce *extra randomness* which may potentially violate the submodularity property of MILT. To see this, Fig. 3d plots the greedy criterion value $\text{MILT}(A_{i-1} \cup t_i^\star) - \text{MILT}(A_{i-1})$ (4) corresponding to selected task $t_i^\star$ vs. round i of active task selection. Since MILT is theoretically submodular, we expect to see a monotonically decreasing curve in Fig. 3d. However, we only observe such a monotonicity with the forward-backward method. This is because the evaluation of (4) involves approximation: It approximates the belief of the meta-parameters through SVGD. However, the naive method performs SVGD by initializing the particles randomly, which introduces randomness such that its approximation of the belief can be inconsistent between successive rounds. In contrast, the forward-backward method always performs belief updates with only one task and thus maintains a consistent belief because the belief updates are highly correlated between successive rounds. Such a "consistency" retains the submodularity of (our approximation of) MILT, which allows our algorithm (Algo. 1) to perform satisfactorily, as implied by its near-optimal performance guarantee.

Table 2: Meta-test *mean squared error* (MSE) over 5 runs with varying no. M of particles on 5-shot Sinusoid regression. MILT is used to select $k = 100$ tasks from $|T| = 1000$ candidate tasks.

	$M = 1$	$M = 3$	$M = 5$
5-shot	0.490	0.454	0.430
10-shot	0.144	0.137	0.129

Table 3: Mean runtime (seconds) on 5-shot Sinusoid regression. MILT is used to select from $|T| = 1000$ candidate tasks.

	forward-backward	naive
$k = 20$	24.5	153
$k = 40$	38.9	293
$k = 60$	54.6	425

5.3 Generalization to Adaptive Task Selection

This work considers the case of *non-adaptive* task selection s.t. k tasks are selected greedily (*one per iteration/round*) and their remaining datasets are acquired/observed only after all the k tasks are selected. For the case of **adaptive task selection** s.t. the remaining dataset $(X_t^r, \mathbf{y}_t^r)$ is acquired immediately after each task t is selected, an immediate improvement we can make to our algorithm is to additionally use this remaining dataset to update the forward particles to obtain a more accurate posterior belief of Θ in line 12 of Algorithm 1. As a result, the meta-test performance can be improved, as shown in Table 4. Note, however, that no performance guarantee is available for the adaptive case. Supposing the meta-learner's data need is time-critical (e.g., budget is available for a limited time) and each acquired remaining dataset can only be released after some time due to regulations, non-adaptive task selection may be preferred.

Table 4: Meta-test MSE or accuracy (%) over 5 runs comparing adaptive vs. non-adaptive task selection.

	non-adaptive	adaptive
5-shot Sinusoid ($k = 100$)	0.430	**0.417**
10-shot Sinusoid ($k = 100$)	0.129	**0.124**
1-shot 5-way Omniglot ($k = 120$)	91.9	**93.2**
1-shot 20-way Omniglot ($k = 250$)	88.6	**89.2**
1-shot 5-way MiniImageNet ($k = 100$)	39.5	**41.1**

6 Related Work

Several works have aimed at combining active learning (Cao *et al.*, 2013; Chen *et al.*, 2012, 2013, 2015; Hoang *et al.*, 2014a,b; Ling *et al.*, 2016; Low *et al.*, 2008, 2009, 2011, 2012, 2014; Nguyen *et al.*, 2021b; Ouyang *et al.*, 2014; Zhang *et al.*, 2016) with meta-learning. The work of Pang *et al.* (2018) has used meta-learning to learn the best active learning criterion for querying different datasets, which differs from our aim of using active learning to select tasks for meta-learning. The algorithms of Finn *et al.* (2018); Yoon *et al.* (2018) have actively selected data points in each task but are not capable of selecting tasks directly. The greedy class-pair based sampling proposed in (Liu *et al.*, 2020) and the probabilistic active meta-learning algorithm proposed in (Kaddour *et al.*, 2020) have different problem settings. In comparison, our problem setting is more general, as discussed in Sec. 3. The work of Luna and Leonetti (2020) has proposed an information-theoretic task selection algorithm for meta-reinforcement learning, which assumes the availability of a validation set that can accurately represent the entire task distribution. Such a strong assumption contradicts the motivation of our active task selection problem, as discussed in Sec. 1.

Most relevant to our work here is that of Chen *et al.* (2021) proposing a variance-based greedy task selection algorithm which does not account for the dependence between latent task vectors (Sec. 3.2). In contrast, our algorithm exploits such a dependence via our proposed MILT criterion and also provides a near-optimal performance guarantee. The work of Yu *et al.* (2019) has proposed to use mutual information in meta-inverse reinforcement learning to enforce the connection between the reward function and a latent contextual variable, which is the first combination of mutual information with meta-learning, but in a different context not related to active task selection.

7 Conclusion

This paper describes a novel active task selection algorithm based on MILT for meta-learning with a near-optimal performance guarantee. A forward-backward method based on our proposed online SVGD is also designed to improve our efficiency. Empirical evaluation on several benchmark datasets have demonstrated the state-of-the-art performance of our algorithm. For future work, we plan to investigate the case of adaptive task selection such that the remaining dataset $(X_t^r, \mathbf{y}_t^r)$ is acquired immediately after task t is selected instead of waiting for the budget of k tasks to be expended.

Acknowledgments

This research is supported by the Singapore Ministry of Education Academic Research Fund Tier 1.

References

Brooks, S., Gelman, A., Jones, G., and Meng, X. (2011). *Handbook of Markov chain Monte Carlo*. CRC Press.

Cao, N., Low, K. H., and Dolan, J. M. (2013). Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In *Proc. AAMAS*, pages 7–14.

Chen, J., Low, K. H., Tan, C. K.-Y., Oran, A., Jaillet, P., Dolan, J. M., and Sukhatme, G. S. (2012). Decentralized data fusion and active sensing with mobile sensors for modeling and predicting spatiotemporal traffic phenomena. In *Proc. UAI*, pages 163–173.

Chen, J., Low, K. H., and Tan, C. K.-Y. (2013). Gaussian process-based decentralized data fusion and active sensing for mobility-on-demand system. In *Proc. RSS*.

Chen, J., Low, K. H., Jaillet, P., and Yao, Y. (2015). Gaussian process decentralized data fusion and active sensing for spatiotemporal traffic modeling and prediction in mobility-on-demand systems. *IEEE Trans. Autom. Sci. Eng.*, **12**, 901–921.

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *Proc. ICML*, pages 1683–1691.

Chen, Y., Li, D., Li, N., Liang, T., Zhang, S., and Low, B. K. H. (2021). Meta-learning with implicit processes. `https://openreview.net/forum?id=m2ZxDprKYlO`.

Eberhard, S. and Paul, R. (2019). A community-owned exchange for health information powered by

blockchain technology. HIT Whitepaper version 4.01, Health Information Traceability Foundation.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, pages 1126–1135.

Finn, C., Xu, K., and Levine, S. (2018). Probabilistic model-agnostic meta-learning. In *Proc. NeurIPS*, pages 9516–9527.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. E. (2019). Meta-learning probabilistic inference for prediction. In *Proc. ICLR*.

Hernández-Lobato, D., Hernández-Lobato, J. M., and Dupont, P. (2011). Robust multi-class Gaussian process classification. In *Proc. NeurIPS*, pages 280–288.

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

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

Kaddour, J., Sæmundsson, S., and Deisenroth, M. P. (2020). Probabilistic active meta-learning. In *Proc. NeurIPS*.

Ko, C., Lee, J., and Queyranne, M. (1995). An exact algorithm for maximum entropy sampling. *Operations Research*, **43**(4), 684–691.

Krause, A. and Golovin, D. (2014). Submodular function maximization. *Tractability*, **3**, 71–104.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, **9**(8), 235–284.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. (2011). One shot learning of simple visual concepts. In *Proc. CogSci*, volume 33.

Ling, C. K., Low, B. K. H., and Jaillet, P. (2016). Gaussian process planning with Lipschitz continuous reward functions: Towards unifying Bayesian optimization, active learning, and beyond. In *Proc. AAAI*, pages 1860–1866.

Liu, C., Wang, Z., Sahoo, D., Fang, Y., Zhang, K., and Hoi, S. (2020). Adaptive task sampling for meta-learning. In *Proc. ECCV*, pages 752–769.

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Proc. NeurIPS*, pages 2378–2386.

Low, K. H., Dolan, J. M., and Khosla, P. (2008). Adaptive multi-robot wide-area exploration and mapping. In *Proc. AAMAS*, pages 23–30.

Low, K. H., Dolan, J. M., and Khosla, P. (2009). Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. In *Proc. ICAPS*, pages 233–240.

Low, K. H., Dolan, J. M., and Khosla, P. (2011). Active Markov information-theoretic path planning for robotic environmental sensing. In *Proc. AAMAS*, pages 753–760.

Low, K. H., Chen, J., Dolan, J. M., Chien, S., and Thompson, D. R. (2012). Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proc. AAMAS*, pages 105–112.

Low, K. H., Chen, J., Hoang, T. N., Xu, N., and Jaillet, P. (2014). Recent advances in scaling up Gaussian process predictive models for large spatiotemporal data. In S. Ravela and A. Sandu, editors, *Dynamic Data-Driven Environmental Systems Science: First International Conference, DyDESS 2014*, pages 167–181. LNCS 8964, Springer International Publishing.

Luna, G. R. and Leonetti, M. (2020). Information-theoretic task selection for meta-reinforcement learning. In *Proc. NeurIPS*.

Ma, C., Li, Y., and Hernández-Lobato, J. M. (2019). Variational implicit processes. In *Proc. ICML*, pages 4222–4233.

Neal, R. M. (1993). Bayesian learning via stochastic dynamics. In *Proc. NeurIPS*, pages 475–482.

Nguyen, Q. P., Low, B. K. H., and Jaillet, P. (2020). Variational Bayesian unlearning. In *Proc NeurIPS*, pages 16025–16036.

Nguyen, Q. P., Low, B. K. H., and Jaillet, P. (2021a). Learning to learn with Gaussian processes. In *Proc UAI*, pages 1466–1475.

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

Ouyang, R., Low, K. H., Chen, J., and Jaillet, P. (2014). Multi-robot active sensing of non-stationary Gaussian process-based environmental phenomena. In *Proc. AAMAS*, pages 573–580.

Pang, K., Dong, M., and Hospedales, T. (2018). Meta-learning transferable active learning policies by deep reinforcement learning. `https://openreview.net/forum?id=HJ4IhxZAb`.

Ravi, S. and Beatson, A. (2018). Amortized Bayesian meta-learning. In *Proc. ICLR*.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *Proc. ICLR*.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *Proc. ICLR*.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. (2018). Bayesian model-agnostic meta-learning. In *Proc. NeurIPS*, pages 7332–7342.

Yu, L., Yu, T., Finn, C., and Ermon, S. (2019). Meta-inverse reinforcement learning with probabilistic context variables. In *Proc. NeurIPS*, pages 11772–11783.

Zhang, Y., Hoang, T. N., Low, K. H., and Kankanhalli, M. (2016). Near-optimal active learning of multi-output Gaussian processes. In *Proc. AAAI*, pages 2351–2357.

Algorithm 2 Near-Optimal Active Task Selection based on MILT (Naive Method without Forward-Backward Method)

1: Set $A = \emptyset$;
2: **while** $|A| < k$ **do**
3: Sample $\mathbf{z}_A \sim p(\mathbf{z}_A)$, $\mathbf{z}_{\overline{A}} \sim p(\mathbf{z}_{\overline{A}})$;
4: **for** $t \in T \setminus A$ **do**
5: Compute $p(\theta|\mathbf{z}_A, \mathbf{y}_A^s) \leftarrow \text{SVGD}_\Theta\left(p(\theta), \{\mathbf{z}_A, \mathbf{y}_A^s\}\right)$;
6: Compute $p\Big(\theta|\mathbf{z}_{\overline{A\cup t}}, \mathbf{y}_{\overline{A\cup t}}^s\Big) \leftarrow \text{SVGD}_\Theta\left(p(\theta), \{\mathbf{z}_{\overline{A\cup t}}, \mathbf{y}_{\overline{A\cup t}}^s\}\right)$;
7: Compute with $p(\theta|\mathbf{z}_A, \mathbf{y}_A^s) : p(\mathbf{z}_t|\mathbf{z}_A) \leftarrow \text{VI}_{\mathcal{Z}}(p(\theta|\mathbf{z}_A, \mathbf{y}_A^s))$;
8: Compute with $p\Big(\theta|\mathbf{z}_{\overline{A\cup t}}, \mathbf{y}_{\overline{A\cup t}}^s\Big) : p(\mathbf{z}_t|\mathbf{z}_{\overline{A\cup t}}) \leftarrow \text{VI}_{\mathcal{Z}}\left(p\Big(\theta|\mathbf{z}_{\overline{A\cup t}}, \mathbf{y}_{\overline{A\cup t}}^s\Big)\right)$;
9: Estimate $H(\mathcal{Z}_t|\mathcal{Z}_A) - H(\mathcal{Z}_t|\mathcal{Z}_{\overline{A\cup t}})$;
10: **end for**
11: Select $t^\star = \arg\max_t H(\mathcal{Z}_t|\mathcal{Z}_A) - H(\mathcal{Z}_t|\mathcal{Z}_{\overline{A\cup t}})$;
12: Update $A = A \cup t^\star$;
13: **end while**
14: **return** A

A Classification with Robust-Max Likelihood

We will discuss the robust-max likelihood for meta-learning of the probabilistic classification tasks here. Following that of Hernández-Lobato *et al.* (2011), the likelihood of a data point $(\mathbf{x}, y_\mathbf{x})$ in a N-way classification problem given an N-dimensional IP output $\mathbf{f} \triangleq [f_1, f_2, \ldots, f_N]$ depends on a binary variable a (i.e., one per $\mathbf{x}$ indicating if $\arg\max \mathbf{f}$ prediction is correct or not):

$$p(y_\mathbf{x}|\mathbf{x}, \mathbf{f}, a) = \prod_{c \neq y_\mathbf{x}} \Theta(f_{y_\mathbf{x}} - f_c)^{1-a} \ (1/N)^a$$

where $\Theta(\cdot)$ is the Heaviside step function and a follows a factorizing multivariate Bernoulli prior distribution:

$$p(a|\rho) \triangleq \text{Bern}(a|\rho) = \rho^a \ (1-\rho)^{1-a}$$

such that ρ estimates the fraction of outliers in the training data. The prior of ρ is defined as a conjugate beta distribution:

$$p(\rho) \triangleq \text{Beta}(\rho|a_0, b_0) = \frac{\rho^{a_0-1} \ (1-\rho)^{b_0-1}}{\text{BetaF}(a_0, b_0)}$$

where $\text{BetaF}(\cdot,\cdot)$ is the beta function, and a_0 and b_0 are free hyperparameters which do not have a big effect on the final model, provided that $b_0 > a_0$ and that they are not too small (Hernández-Lobato *et al.*, 2011). We follow that of Chen *et al.* (2021) to set $a_0 = 1$ and $b_0 = 9$.

B Computational Cost of Forward-Backward Method and Naive Method

We will analyze here the computational cost of both the forward-backward method and the naive method to understand the improvement by adopting the former. Algo. 1 describes the forward-backward method, while Algo. 2 presents the naive method. The difference lies mainly in lines 7 and 8 of Algo. 1 where the forward-backward method uses online SVGD to perform fast belief updates for the selected task. On the other hand, the naive method computes such a belief of the meta-parameters from scratch using A (and $T \setminus (A \cup t)$) in lines 5 and 6 of Algo. 2.

Though the computational cost of VI (containing multiple SVGD iterations till convergence) on n tasks (specifically, the n sample datasets) is not necessarily n times the computational cost of VI on 1 task, we have observed in practice that VI on a *minibatch* of tasks converges within (slightly more but) nearly the same number of iterations as VI on one task; both require around 5 SVGD iterations. So, we have considered the following assumption:

Assumption 1 (informal). *Suppose that the GPU can afford to process a minibatch of $n \leq B$ tasks in parallel. VI (using SVGD to compute posterior belief of the meta-parameters) on a minibatch of $n \leq B$ tasks converges in the same number of iterations as VI on 1 task.*

Note that line 3 of Algo. 1 and lines 5 and 6 of Algo. 2 can enjoy computational benefits from such parallel processing of each minibatch of tasks. Since (a) the overall computational cost is dominated by the number of SVGD iterations in the algorithms and (b) the computational cost of one iteration of online SVGD (for both learning or unlearning) is the same as that of one SVGD iteration per task, the following results ensue:

Remark 1. *The computational cost measured by the number of floating point operations is $\mathcal{O}(k|T|)$ for the forward-backward method and $\mathcal{O}(k|T|^2)$ for the naive method.*

Remark 2. *The computational cost measured by the runtime is $\mathcal{O}(k|T|)$ for the forward-backward method and $\mathcal{O}(k|T|^2/B)$ for the naive method.*

The above results arise from iterating through all candidate tasks in T in each of the k rounds of Algo. 1 or Algo. 2. For every candidate task, the naive method requires $\mathcal{O}(|T|)$ floating point operations in SVGD, while the forward-backward method requires only $\mathcal{O}(1)$ floating point operations in online SVGD.

As can be concluded from the above remarks, the forward-backward method improves time efficiency by $|T|$ times over the naive method because its online SVGD performs fast belief updates for only 1 selected task instead of using A (and $T \setminus (A \cup t)$). The parallel processing of each minibatch of tasks can accelerate the naive method by B times, but the naive method is still much slower than the forward-backward method since usually, $B \ll |T|$.

C EM Algorithm for Meta-Learning and Modified Variant

C.1 Probabilistic Meta-Learning with IP

Meta-learning on a set A of tasks, which adopt a split of their corresponding datasets according to Sec. 2, can be defined as the inference of meta-parameters Θ with the following joint likelihood to be maximized (Chen *et al.*, 2014; Finn *et al.*, 2017, 2018):

$$p(\mathbf{y}_A^r, \theta|\mathbf{y}_A^s) = p(\theta)\prod_{t\in A} p(\mathbf{y}_t^r|\theta,\mathbf{y}_t^s) = p(\theta)\prod_{t\in A}\int_{\mathbf{f}_t^r} p(\mathbf{y}_t^r|\mathbf{f}_t^r)\; p(\mathbf{f}_t^r|\theta,\mathbf{y}_t^s)\; \mathrm{d}\mathbf{f}_t^r. \tag{10}$$

Task *adaptation* $p(\mathbf{f}_t^r|\theta,\mathbf{y}_t^s)$ is performed via IP inference (Chen *et al.*, 2021) given the sample dataset $(X_t^s,\mathbf{y}_t^s)$:

$$p(\mathbf{f}_t^r|\theta,\mathbf{y}_t^s) = \int_{\mathbf{z}} p(\mathbf{f}_t^r|\mathbf{z},\theta)\; p(\mathbf{z}|\theta,\mathbf{y}_t^s)\; \mathrm{d}\mathbf{z}\;. \tag{11}$$

Fig. 1 shows a graphical model of the IP for meta-learning. Note that in meta-learning, we have to model the belief of $\mathcal{Y}_t^r$ and use both the sample and remaining datasets to perform meta-training. On the other hand, we have to model the belief of $\mathcal{Y}_t^s$ to perform task selection (Sec. 4).

Chen *et al.* (2021) have used an *expectation maximization* (EM) algorithm to perform meta-training such that the E step carries out the IP inference of $\mathcal{Z}$ (and $f(\cdot)$) in (11) and the M step maximizes the joint likelihood (10) w.r.t. a point estimate of Θ. Note that with our additional Bayesian treatment of Θ, meta-training returns a posterior belief $p(\theta|\mathbf{y}_A^s,\mathbf{y}_A^r)$ of the meta-parameters instead of a point estimate, which is empirically shown to improve performance (Sec. 5). While Chen *et al.* (2021) are mainly interested in deriving such an EM algorithm for meta-training, the motivation of our work here is to derive an efficient active task selection algorithm for meta-learning with a near-optimal performance guarantee. Nevertheless, we have modified their EM algorithm to work in our case with the Bayesian treatment of Θ.

C.2 Our modification

We will provide details of the EM algorithm proposed by Chen *et al.* (2021) and describe our modifications here.

Expectation (E) step. The aim of the E step is to obtain (samples of) $p(\mathbf{f}_t^r|\theta,\mathbf{y}_t^s)$. Note that (samples of) $p(\mathbf{f}_t^r|\theta,\mathbf{y}_t^s)$ can be obtained using the generator g_Θ (1) and the *latent task posterior belief* $p(\mathbf{z}|\theta,\mathbf{y}_t^s)$, as follows: First draw samples of $\mathbf{z}$ from $p(\mathbf{z}|\theta,\mathbf{y}_t^s)$, and then passing them and X_t^r as inputs to generator g_Θ to obtain samples from $p(\mathbf{f}_t^r|\theta,\mathbf{y}_t^s)$. Hence, for a task t, performing its adaptation $p(\mathbf{f}_t^r|\theta,\mathbf{y}_t^s)$ (11) reduces to obtaining the latent task posterior belief $p(\mathbf{z}|\theta,\mathbf{y}_t^s)$.

In general, $p(\mathbf{z}|\theta,\mathbf{y}_t^s)$ cannot be evaluated in closed form. Instead of using *variational inference* (VI), the work of Chen *et al.* (2021) has drawn samples from $p(\mathbf{z}|\theta,\mathbf{y}_t^s)$ using *stochastic gradient Hamiltonian Monte Carlo*

(SGHMC), which introduces an auxiliary random vector $\mathbf{r}$ and samples from the joint distribution $p(\mathbf{z}, \mathbf{r}|\theta, \mathbf{y}_t^s)$ following the Hamiltonian dynamics (Brooks *et al.*, 2011; Neal, 1993) and making use of the tractable gradient on $\mathcal{Z}$: $-\nabla_{\mathbf{z}} \log p(\mathbf{z}|\theta, \mathbf{y}_t^s) = -\nabla_{\mathbf{z}} \log p(\mathbf{z}, \mathbf{y}_t^s|\theta) = -\nabla_{\mathbf{z}}[\log p(\mathbf{y}_t^s|\mathbf{f}_t^s = (g_\theta(\mathbf{x}, \mathbf{z}))_{\mathbf{x}\in X_t^s}^\top) + \log p(\mathbf{z})]$.

Our modification of E step: We represent the belief of Θ as a set $\{\theta^m\}_{m=1}^M$ of M particles, which we denote as $q(\theta)$ without loss of generality. To accelerate computation, we use VI in Sec. 4.2 to compute $p(\mathbf{z}|\theta^m, \mathbf{y}_t^s)$:

$$p(\mathbf{z}|\theta^m, \mathbf{y}_t^s) \leftarrow \mathrm{VI}_{\mathcal{Z}}(\theta^m) \ .$$

We then pass the samples of $p(\mathbf{z}|\mathbf{y}_t^s, \theta^m)$ to the generator g_{θ^m} (i.e., corresponding to the particle θ^m) to obtain samples of $p(\mathbf{f}_t^r|\theta^m, \mathbf{y}_t^s)$.

Maximization (M) step. The aim of the M step is to optimize (10) w.r.t. a point estimate of θ using the samples from the E step. In particular, (10) is optimized through gradient descent w.r.t. θ using samples of $\mathbf{z}$ from the E step.

Our modification of M step: Instead of using gradient descent, we use SVGD (Sec. 4.1) on the datasets $(X_A, \mathbf{y}_A)$ of the subset A of tasks to compute $p(\theta|\mathbf{y}_A)$ such that in every SVGD iteration, each particle θ^m is updated as follows (i.e., similar to that in Sec. 4.1):

$$\theta^m \leftarrow \theta^m + \frac{\eta}{M} \sum_{\theta\in\{\theta^m\}_{m=1}^M} \big[k(\theta, \theta^m) \\ \times \nabla_\theta \log p(\mathbf{y}_A^r|\theta, \mathbf{y}_A^s) + \nabla_\theta k(\theta, \theta^m)\big]$$

where η is the step size, and $k(\cdot, \cdot)$ is a *radial basis function* kernel representing a repulsive force between particles to prevent them from collapsing. We denote the entire VI process (containing multiple SVGD iterations till convergence) to obtain $p(\theta|\mathbf{y}_A)$ as

$$p(\theta|\mathbf{y}_A) \leftarrow \mathrm{SVGD}_\Theta(p(\theta), \mathbf{y}_A) \ .$$

D Proofs and other Theoretical Results

D.1 Proof of Theorem 1

Firstly, note that the MILT$(\cdot)$ function (2) is submodular:

Lemma 1. *The set function $A \mapsto \mathrm{MILT}(A)$ is submodular.*

Its proof can be found in (Krause and Golovin, 2014). We can also formally prove that the MILT$(\cdot)$ function (2) is approximately monotonic up to a constant error of C_0:

Lemma 2. $\forall \mathcal{B} \subseteq T \setminus A \ \ \mathrm{MILT}(A \cup \mathcal{B}) \geq \mathrm{MILT}(A) - C_0$ *where* $C_0 \triangleq H(\Theta)$.

Proof.

$$\begin{aligned}
&\mathrm{MILT}(A \cup \mathcal{B}) - \mathrm{MILT}(A) \\
&= H(\mathcal{Z}_\mathcal{B}|\mathcal{Z}_A) - H(\mathcal{Z}_\mathcal{B}|\mathcal{Z}_{T\setminus(A\cup\mathcal{B})}) \\
&\geq H(\mathcal{Z}_\mathcal{B}|\mathcal{Z}_A, \Theta) - H(\mathcal{Z}_\mathcal{B}|\mathcal{Z}_{T\setminus(A\cup\mathcal{B})}) \\
&\geq H(\mathcal{Z}_\mathcal{B}|\mathcal{Z}_A, \Theta) - H(\mathcal{Z}_\mathcal{B}, \Theta|\mathcal{Z}_{T\setminus(A\cup\mathcal{B})}) \\
&= H(\mathcal{Z}_\mathcal{B}|\Theta) - H(\mathcal{Z}_\mathcal{B}, \Theta|\mathcal{Z}_{T\setminus(A\cup\mathcal{B})}) \\
&\geq H(\mathcal{Z}_\mathcal{B}|\Theta) - (H(\mathcal{Z}_\mathcal{B}|\Theta, \mathcal{Z}_{T\setminus(A\cup\mathcal{B})}) + H(\Theta|\mathcal{Z}_{T\setminus(A\cup\mathcal{B})})) \\
&= -H(\Theta|\mathcal{Z}_{T\setminus(A\cup\mathcal{B})}) \\
&\geq -H(\Theta)
\end{aligned}$$

where the first equality is by definition of MILT$(\cdot)$ function (2), the first and last inequalities are due to the "conditioning reduces entropy" property, the second and third inequalities are due to chain rule for entropy, and the second and last equalities follow from $\mathcal{Z}_t, \mathcal{Z}_{t'}, \forall t \neq t'$ being mutually independent given meta-parameters Θ. □

Proof of Theorem 1. Let $t_1^\star,\ldots,t_k^\star$ be the k tasks selected by Algo. 1. So, $A_i=\{t_1^\star,\ldots,t_i^\star\}$. Recall from (3) that $A^\star \triangleq \arg\max_{A\subseteq T,|A|=k}\text{MILT}(A)$. From Lemma 2,

$$\text{MILT}(A^\star\cup A_i)\geq \text{MILT}(A^\star)-C_0 \tag{12}$$

for all rounds $i=1,\ldots,k$. Let $\Delta_i \triangleq \text{MILT}(A_i)-\text{MILT}(A_{i-1})$. From Lemma 1, we know that for any A and $t\notin(A\cup A_i)$,

$$\begin{aligned}&\text{MILT}((A\cup A_i)\cup t)-\text{MILT}(A\cup A_i)\\ &\leq \text{MILT}(A_i\cup t)-\text{MILT}(A_i)\leq \Delta_{i+1}\ .\end{aligned}$$

It follows that for any A s.t. $|A|=k$,

$$\text{MILT}(A\cup A_i)-\text{MILT}(A_i)\leq k\Delta_{i+1}\ .$$

Consequently,

$$\text{MILT}(A^\star\cup A_i)\leq \text{MILT}(A_i)+k\Delta_{i+1}=\sum_{j=1}^{i}\Delta_j+k\Delta_{i+1}\ . \tag{13}$$

From (12) and (13),

$$\text{MILT}(A^\star)-C_0\leq\sum_{j=1}^{i}\Delta_j+k\Delta_{i+1} \tag{14}$$

for round $i=0,\ldots,k-1$. We refer to (14) as the i-th inequality. Now, by multiplying both sides of the i-th inequality by a factor of $(1/k)(1-(1/k))^{k-i-1}$ and then summing both sides over all rounds $i=0,\ldots,k-1$,

$$\begin{aligned}&(\text{MILT}(A^\star)-C_0)\sum_{i=0}^{k-1}\frac{1}{k}\left(1-\frac{1}{k}\right)^{k-i-1}\\ &\leq\sum_{i=1}^{k}\Delta_i=\text{MILT}(A_k)\ .\end{aligned} \tag{15}$$

To understand why the coefficient of Δ_i in (15) reduces to 1 for $i=1,\ldots,k$, the Δ_i term in the j-th inequality (14) for $j=i,\ldots,k-1$ has a coefficient of $(1/k)(1-(1/k))^{k-j-1}$, while the $k\Delta_i$ term in the $(i-1)$-th inequality (14) has a coefficient of $(1/k)(1-(1/k))^{k-(i-1)-1}$. Then,

$$\begin{aligned}&\text{coefficient of }\Delta_i\text{ in (15)}\\ &=1\times\sum_{j=i}^{k-1}\frac{1}{k}\left(1-\frac{1}{k}\right)^{k-j-1}+k\times\frac{1}{k}\left(1-\frac{1}{k}\right)^{k-(i-1)-1}\\ &=1\ .\end{aligned}$$

From (15),

$$\begin{aligned}\text{MILT}(A_k)&\geq(\text{MILT}(A^\star)-C_0)\left(1-\left(1-\frac{1}{k}\right)^k\right)\\ &\geq(\text{MILT}(A^\star)-C_0)(1-1/e)\ .\end{aligned}$$

D.2 Near-Optimal Performance Guarantee for MIRD

We will now describe a greedy algorithm similar to Algo. I:

Algorithm II. *Start with an empty set $A_0=\emptyset$ of tasks. In each round $i=1,\ldots,k$, greedily select the next task:*

$$t_i^\star\triangleq\arg\max_t\text{MIRD}(A_{i-1}\cup t)-\text{MIRD}(A_{i-1}) \tag{16}$$

and update the set $A_i=A_{i-1}\cup t_i^\star=\{t_1^\star,\ldots,t_i^\star\}$ of selected tasks.

Theorem 2 (Near-optimal performance guarantee)**.** *Algorithm II is guaranteed to select a set A of k tasks s.t.*

$$\text{MIRD}(A)\geq(1-1/e)(\text{OPT}-C_0)$$

where $\text{OPT}\triangleq\max_{A\subseteq T,|A|=k}\text{MIRD}(A)$ *and the constant $C_0=H(\Theta)$ is the entropy of meta-parameters Θ.*

Lemma 3. $\forall \mathcal{B} \subset T \setminus A \;\; \mathit{MIRD}(A \cup \mathcal{B}) \geq \mathit{MIRD}(A) - C_0$ *where* $C_0 \triangleq H(\Theta)$.

Proof.

$$\begin{aligned}
&\mathrm{MIRD}(A \cup \mathcal{B}) - \mathrm{MIRD}(A)\\
&= H(\mathcal{Y}^r_{\mathcal{B}}|\mathcal{Y}^r_A) - H(\mathcal{Y}^r_{\mathcal{B}}|\mathcal{Y}^r_{T\setminus(A\cup\mathcal{B})})\\
&\geq H(\mathcal{Y}^r_{\mathcal{B}}|\mathcal{Y}^r_A,\Theta) - H(\mathcal{Y}^r_{\mathcal{B}}|\mathcal{Y}^r_{T\setminus(A\cup\mathcal{B})})\\
&\geq H(\mathcal{Y}^r_{\mathcal{B}}|\mathcal{Y}^r_A,\Theta) - H(\mathcal{Y}^r_{\mathcal{B}},\Theta|\mathcal{Y}^r_{T\setminus(A\cup\mathcal{B})})\\
&= H(\mathcal{Y}^r_{\mathcal{B}}|\Theta) - H(\mathcal{Y}^r_{\mathcal{B}},\Theta|\mathcal{Y}^r_{T\setminus(A\cup\mathcal{B})})\\
&\geq H(\mathcal{Y}^r_{\mathcal{B}}|\Theta) - (H(\mathcal{Y}^r_{\mathcal{B}}|\Theta,\mathcal{Y}^r_{T\setminus(A\cup\mathcal{B})}) + H(\Theta|\mathcal{Y}^r_{T\setminus(A\cup\mathcal{B})}))\\
&= -H(\Theta|\mathcal{Y}^r_{T\setminus(A\cup\mathcal{B})})\\
&\geq -H(\Theta)
\end{aligned}$$

where the first equality is by definition of MILT$(\cdot)$ function (2), the first and last inequalities are due to the "conditioning reduces entropy" property, the second and third inequalities are due to chain rule for entropy, and the second and last equalities follow from $\mathcal{Y}^r_t, \mathcal{Y}^r_{t'}, \forall t \neq t'$ being mutually independent given meta-parameters Θ. □

Proof of Theorem 2. Similar to MILT, MIRD is submodular since it is also a mutual information criterion. Lemma 3 reveals that MIRD is approximately monotonic. Therefore, we can adopt the same proof as that for MILT (Appendix D.1) to derive the near-optimal performance guarantee for MIRD.

D.3 Online SVGD for Learning: Theoretical Result

Proposition 2 (**Online SVGD for learning**). *Suppose that $p(\theta)$ is an uninformative prior (e.g., $\nabla_\theta \log p(\theta) = 0$). With $\{\theta^m\}_{m=1}^M$ initially sampled from $p(\theta|\mathbf{z}_A, \mathbf{y}^s_A)$, the SVGD operation for obtaining $p(\theta|\mathbf{z}_{A\cup t}, \mathbf{y}^s_{A\cup t})$ (i.e., learning from a task $t \notin A$) is*

$$\begin{aligned}
\theta^m \leftarrow \theta^m + \frac{\eta}{M} \sum_{\theta\in\{\theta^m\}_{m=1}^M} \big[k(\theta,\theta^m)\\
\times\nabla_\theta \log p(\theta|\mathbf{z}_t,\mathbf{y}^s_t) + \nabla_\theta k(\theta,\theta^m)\big] \;.
\end{aligned} \tag{17}$$

We denote such a learning process (containing multiple SVGD iterations till convergence) as

$$p(\theta|\mathbf{z}_{A\cup t}, \mathbf{y}^s_{A\cup t}) \leftarrow \mathit{SVGD}_\Theta(p(\theta|\mathbf{z}_A, \mathbf{y}^s_A), \{\mathbf{z}_t, \mathbf{y}^s_t\}) \;.$$

Proof. With $\{\theta^m\}_{m=1}^M$ initially sampled from an arbitrary distribution $q(\theta)$, the SVGD operation for learning $p(\theta|\mathbf{z}_{A\cup t}, \mathbf{y}^s_{A\cup t})$ is

$$\begin{aligned}
\theta^m &\leftarrow \theta^m + \frac{\eta}{M} \sum_{\theta\in\{\theta^m\}_{m=1}^M} \big[k(\theta,\theta^m)\\
&\qquad \times\nabla_\theta \log p(\theta|\mathbf{z}_{A\cup t},\mathbf{y}^s_{A\cup t}) + \nabla_\theta k(\theta,\theta^m)\big]\\
&= \theta^m + \frac{\eta}{M} \sum_{\theta\in\{\theta^m\}_{m=1}^M} \big[k(\theta,\theta^m)\\
&\qquad \times\nabla_\theta (\log p(\theta|\mathbf{z}_t,\mathbf{y}^s_t) + \log p(\theta|\mathbf{z}_A,\mathbf{y}^s_A) - \log p(\theta))\\
&\qquad +\nabla_\theta k(\theta,\theta^m)\big] \;.
\end{aligned}$$

Note that SVGD is a discretization of the underlying continuous functional gradient in the *reproducing kernel Hilbert space* (RKHS) of $k(\cdot,\cdot)$. The corresponding continuous functional gradient, which is proven to be the expectation of the Stein's operator (Liu and Wang, 2016), of the above SVGD is

$$\begin{aligned}
\mathbb{E}_{q(\theta)}[k(\theta,\cdot) \times \nabla_\theta \big(&\log p(\theta|\mathbf{z}_t,\mathbf{y}^s_t)\\
&+ \log p(\theta|\mathbf{z}_A,\mathbf{y}^s_A) - \log p(\theta)\big) + \nabla_\theta k(\theta,\cdot)] \;.
\end{aligned}$$

Since we have assumed an uninformative prior $p(\theta)$, $\nabla_\theta \log p(\theta) = 0$ in the above SVGD. Also, using Stein's identity (Liu and Wang, 2016), $\mathbb{E}_{p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)}[k(\theta,\cdot)\nabla_\theta \log p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)] = 0$. Since $q(\theta) = p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)$, the above functional gradient reduces to

$$\mathbb{E}_{q(\theta)}[k(\theta,\cdot)\nabla_\theta \log p(\theta|\mathbf{z}_t,\mathbf{y}_t^s) + \nabla_\theta k(\theta,\cdot)]$$

which results in the SVGD in (17). □

D.4 Proof of Proposition 1

Unlearning from a task $t \in A$ can be cast as a problem of minimizing the *evidence upper bound* (EUBO):

$$\text{EUBO} \triangleq \mathbb{E}_{q(\theta)}[\log p(\mathbf{y}_t^s|\mathbf{z}_t,\theta)] + \text{KL}(q(\theta)\|p(\theta|\mathbf{z}_A,\mathbf{y}_A^s))$$

which yields the exact solution to the original problem of maximizing the *evidence lower bound* (ELBO), which involves learning from the set $A \setminus t$ of tasks (Nguyen *et al.*, 2020).

The above problem can also be cast as one of maximizing the *negative EUBO* (NEUBO): w.r.t. $q(\theta)$:

$$\text{NEUBO} \triangleq \mathbb{E}_{q(\theta)}[-\log p(\mathbf{y}_t^s|\mathbf{z}_t,\theta)] - \text{KL}(q(\theta)\|p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)) \ . \tag{18}$$

Note that SVGD is a discretization of the underlying continuous functional gradient in the *reproducing kernel Hilbert space* (RKHS) of $k(\cdot,\cdot)$. The corresponding continuous functional gradient, which is proven to be the expectation of the Stein's operator (Liu and Wang, 2016), of $-\text{KL}(q(\theta)\|p(\theta|\mathbf{y}_A^s,\mathbf{z}_A))$ is

$$\mathbb{E}_{q(\theta)}[k(\theta,\cdot)\nabla_\theta\big(\log p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)\big) + \nabla_\theta k(\theta,\cdot)] \ .$$

Using Stein's identity (Liu and Wang, 2016), $\mathbb{E}_{p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)}[k(\theta,\cdot)\nabla_\theta \log p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)] = 0$. Since $q(\theta) = p(\theta|\mathbf{z}_A,\mathbf{y}_A^s)$, the above functional gradient reduces to $\mathbb{E}_{q(\theta)}[\nabla_\theta k(\theta,\cdot)]$.

On the other hand, the functional gradient of $\mathbb{E}_{q(\theta)}[-\log p(\mathbf{y}_t^s|\mathbf{z}_t,\theta)]$ (Liu and Wang, 2016) is

$$\begin{aligned}&-\mathbb{E}_{q(\theta)}[k(\theta,\cdot)\nabla_\theta(\log p(\theta|\mathbf{z}_t,\mathbf{y}_t^s) - \log p(\theta))]\\ &= -\mathbb{E}_{q(\theta)}[k(\theta,\cdot)\nabla_\theta \log p(\theta|\mathbf{z}_t,\mathbf{y}_t^s)]\end{aligned}$$

such that the equality follows from our assumption of an uninformative prior $p(\theta)$, that is, $\nabla_\theta \log p(\theta) = 0$. By summing the above two functional gradients, we obtain the functional gradient of (18):

$$\mathbb{E}_{q(\theta)}[-k(\theta,\cdot)\nabla_\theta \log p(\theta|\mathbf{z}_t,\mathbf{y}_t^s) + \nabla_\theta k(\theta,\cdot)]$$

which results in the SVGD in (9).

E More Experimental Results

E.1 Some Implementation Details

For Sinusoid regression, we execute 1 SVGD iteration to perform the adaptation of a task in meta-training and execute 10 SVGD iterations to perform the adaptation of a task in meta-test. The training is performed for a total of 15000 iterations with an Adam optimizer. For Omniglot and MiniImagenet classification, we execute 5 SVGD iterations to perform the adaptation of a task in meta-training and execute 10 SVGD iterations to perform the adaptation of a task in meta-test. The training is performed for a total of 60000 iterations with an Adam optimizer. For other meta-training settings, we adopt the same as that in (Chen *et al.*, 2021).

E.2 Comparison with Two Recent Baselines: Probabilistic Active Meta-learning algorithm (Kaddour *et al.*, 2020) and Greedy Class-Pair Based Sampling (Liu *et al.*, 2020)

The following describes our implementation details of the two baselines which we compared with. Note that both works have different problem settings as compared to ours. Thus, we have to adapt their implementation to perform a fair empirical comparison based on the experimental settings discussed in our paper (Sec. 5).

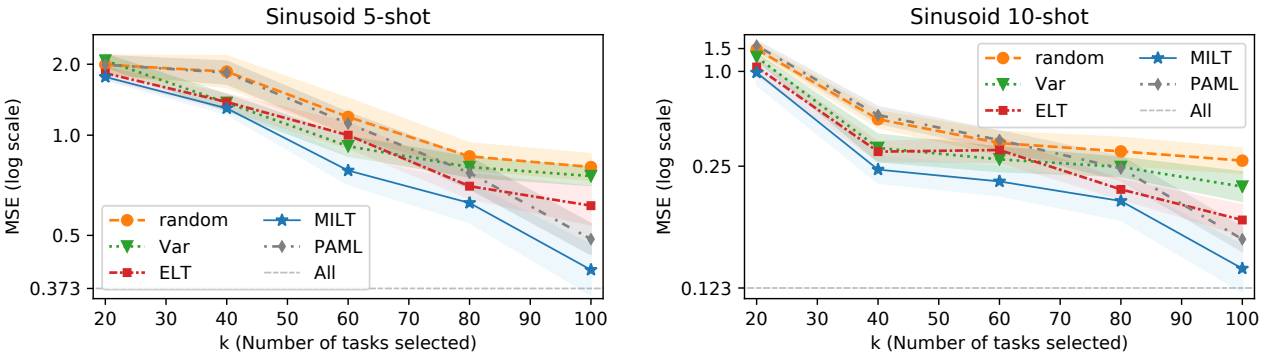

Figure 5: Meta-test *mean squared error* (MSE) and standard error over 5 runs vs. no. k of selected tasks on (left) 5-shot Sinusoid and (right) 10-shot Sinusoid.

E.2.1 Probabilistic active meta-learning algorithm (PAML)

To describe our adapted implementation of PAML from (Kaddour *et al.*, 2020), contextual information is omitted to ensure a fair comparison since all other baselines in our experiments do not need to assume the availability of such contextual information. To achieve this, we have removed the regularization term associated with the contextual information (i.e., the summation term in equation 10 of Kaddour *et al.* (2020)). Then, the meta-training objective becomes a typical meta-learning objective (i.e., equation 5 of Kaddour *et al.* (2020)) which is similar to (i.e., a variational lower bound of) ours that explicitly represents the distributions via variational parameters. In our implementation of PAML, we have adopted the same Gaussian variational distribution of the latent vector in (Kaddour *et al.*, 2020) and used the same model architectures as the ones in our baselines.

Results and discussion. The results on Sinusoid are presented in Fig. 5 and that on Omniglot and MiniImageNet are presented in Fig. 6. The results show that our proposed MILT outperforms PAML. From our experiments, we have observed that PAML is more inclined to select "boundary" tasks: For example, in 10-shot Sinusoid where the amplitude amp is sampled from $[0.1, 5.0]$, PAML has a proportion of 28.9% and 13.3% of its selected tasks from amp > 4.0 and amp < 1.0, respectively. In contrast, our proposed MILT has a proportion of 17.5% and 7.5% of its selected tasks from amp > 4.0 and amp < 1.0, respectively. So, the PAML criterion tends to rank "boundary" tasks more highly due to their greater surprisals (Kaddour *et al.*, 2020). However, selecting too large a proportion of such "boundary" tasks may not yield as much information on the unobserved tasks, which explains why PAML is outperformed by our proposed MILT.

E.2.2 Greedy class-pair based sampling (GCP)

The GCP proposed in (Liu *et al.*, 2020) is an active task selection method for meta-classification tasks. GCP does not directly sample tasks. Instead, it samples classes and then generates the tasks by randomly picking images of the selected classes, which is fundamentally different from our setting. So, to establish a meaningful empirical comparison with GCP, we have to adapt it such that it fits into the general problem setting where the candidate tasks are given/fixed prior to task selection.

In the original GCP setting involving a N-way classification problem, a N-clique of classes is sampled in each round such that the selection probability is formulated based on its potential which is the product of pairwise potentials between two classes.

In our adapted implementation of GCP, we use GCP to compute the potential of each candidate task where each task corresponds to a N-way classification problem. Then, we sample tasks (i.e., one task per round of task selection without replacement) according to the probability computed from the potential. By doing so, instead of allowing GCP to generate tasks, we use the GCP criterion to sample from the given candidate tasks. We adopt the default hyperparameters value (i.e., aggressiveness of 1 and discounting factor of 0.5) in the original GCP paper.

Results and discussion. The results on Omniglot and MiniImageNet are presented in Fig. 6. The results

Table 5: Meta-test *mean squared error* (MSE) and standard deviation over 5 runs on Sinusoid regression.

Batch size:	1	5	10
5-shot	0.39 ± 0.16	0.46 ± 0.14	0.43 ± 0.10
10-shot	0.12 ± 0.03	0.12 ± 0.03	0.13 ± 0.03

Table 6: Meta-test accuracy (%) and standard deviation over 5 runs on Omniglot classification.

Batch size:	16	32	64
1-shot 20-way	89.1 ± 2.9	88.6 ± 3.0	88.8 ± 3.4

Table 7: Meta-test *mean squared error* (MSE) and standard deviation over 5 runs on Sinusoid regression.

No. of candidate tasks:	100	500	1000	2000
5-shot Sinusoid, $k = 100$	0.77 ± 0.19	0.48 ± 0.09	0.43 ± 0.10	0.40 ± 0.07
10-shot Sinusoid, $k = 100$	0.27 ± 0.10	0.19 ± 0.03	0.13 ± 0.03	0.11 ± 0.03

Table 8: Meta-test accuracy (%) and standard deviation over 5 runs.

No. of candidate tasks:	1000	2000	4000
1-shot 5-way Omniglot, $k = 120$	91.0 ± 3.3	91.9 ± 3.0	91.8 ± 3.0
1-shot 20-way Omniglot, $k = 250$	87.5 ± 2.6	88.6 ± 3.0	88.6 ± 2.6

show that our proposed MILT outperforms GCP. We think the reason is that GCP is designed to achieve the best generalization performance on new classes during test time. However, MILT is designed to achieve the best generalization performance on new tasks (but not necessarily new classes) during test time. MILT better fits our general problem setting and thus achieves a better performance than GCP.

E.3 Performance Sensitivity to Number of Sub-tasks in a Task

In Sec. 5, we have mentioned that for Omniglot and MiniImageNet classifications, we have selected tasks in a *batch* manner: Each task consists of 32 sub-tasks s.t. each sub-task is a 1-shot 5-way or 1-shot 20-way classification. We refer to the number of sub-tasks in a task as the *batch size*. For this *batch* setting, the active task selection criterion is the sum of the criterion over all sub-tasks (see footnote 14). Note that such a summation is exact if we know *a priori* that all the sub-tasks in task t correspond to the same latent task vector $\mathcal{Z}_t$. Nevertheless, we will investigate here whether the meta-test performance is sensitive to varying batch sizes.

For Sinusoid regression, we fix the total number of (5-shot or 10-shot) sub-tasks to be selected as 100 (e.g., when the batch size is 10, $k = 10$) and the total number of sub-tasks in the candidate tasks to be 1000.[15] For 1-shot 20-way Omniglot classification, we fix the total number of sub-tasks to be selected as 4000 and the total number of sub-tasks in the candidate tasks to be 64000.

Tables 5 and 6 show results of Sinusoid regression and Omniglot classification, respectively. It can be observed that the meta-test performance of MILT is similar across (and hence not so sensitive to) varying batch sizes which are small compared to the total number of sub-tasks to be selected.

E.4 Performance Sensitivity to Number of Candidate Tasks

In Sec. 5, we have mentioned that we have generated a limited number of candidate tasks (i.e., 1000 for Sinusoid regression and 2000 for Omniglot classification). We have argued in Sec. 5 that we did not generate a massive number (≈ 200000) of candidate tasks since (a) it will require much more computation (linear in $|T|$, as shown in Appendix B) to perform task selection and (b) only a limited number (≤ 250) of tasks will be selected. We will investigate here whether the meta-test performance is sensitive to varying numbers of candidate tasks.

Tables 7 and 8 show results of Sinusoid regression and Omniglot classification, respectively. It can be observed that increasing the number of candidate tasks tends to improve the meta-test performance. However, when the number of candidate tasks becomes relatively large (respectively, 1000 and 2000 for Sinusoid and Omniglot), the improvement in meta-test performance by further increasing the number of candidate tasks is marginal. This

[15] Doing so fixes the number of 5-shot or 10-shot sub-tasks that are available for selection.

empirically supports our argument that our active task selection algorithm may not need such a massive number of candidate tasks, like in previous works (Chen *et al.*, 2021; Finn *et al.*, 2017, 2018; Yoon *et al.*, 2018), in order to achieve reasonably competitive meta-test performance.

E.5 Generalization to Larger Remaining Sets

We have also performed experiments on using a larger remaining dataset to evaluate the meta-test performance with varying ratios of remaining vs. sample dataset sizes from 1 to 20 given a fixed number R of data points in the remaining datasets of all selected tasks. Table 9 shows that the meta-test performance degrades gracefully with an increasing ratio due to a reduced diversity of selected tasks. This reveals that in the setting of meta-learning, the meta-training task diversity plays an important role on the final meta-test performance.

Table 9: Meta-test MSE over 5 runs with varying ratios of remaining vs. sample dataset sizes given a fixed number R of data points in the remaining datasets of all selected tasks.

Ratio	1	5	8	10	20
5-shot Sinusoid ($R = 1000$)	**0.294**	0.315	0.349	0.375	0.398
10-shot Sinusoid ($R = 2000$)	**0.068**	0.070	0.070	0.108	0.129

E.6 Comparison with Information-Theoretic Task Selection (Luna and Leonetti, 2020)

While it is complicated to generalize the existing active task selection algorithms to cater to our more general problem setting (i.e., without making strong assumptions of the availability of contextual information or a validation set (Sec. 6)), we have found a way to adapt the problem setting of the *information-theoretic task selection* (ITTS) algorithm (Luna and Leonetti, 2020) to do so. A core assumption of ITTS is the availability of a validation set that can accurately represent the entire (meta-test) task distribution. Such a strong assumption contradicts the motivation of our active task selection problem, as discussed in Sec. 1. However, we can first construct such a validation set by randomly selecting $k' < k$ candidate tasks as the validation set which we now denote as V.[16] Then, we actively select the rest of the $(k - k')$ tasks according to the ITTS criterion. In our experiments, we set $k' = k/2$.

Note that ITTS is conventionally designed for meta-reinforcement learning, but can be adapted to cater to supervised meta-learning in our problem setting. The two key components of ITTS are (a) the quantification of the *difference* between two tasks measured by KL divergence, and (b) the quantification of the *relevance* of a task t to a task t' measured by the difference in entropy. We adapt their meta-reinforcement learning framework into our (probabilistic) supervised meta-learning framework, as follows:

(a) In the work of Luna and Leonetti (2020), the *difference* between two tasks t and t' is measured as the averaged KL divergence between their respective policies over the states of the validation tasks in V. For supervised meta-learning, it is natural to consider the averaged KL divergence between their latent task beliefs over the latent task vectors representing the tasks in V:

$$\delta(t, t') = \mathbb{E}_{\mathbf{z}_V \sim p(\mathbf{z}_V)}[\mathrm{KL}(p(\mathcal{Z}_t = \mathbf{z}|\mathbf{z}_V) \| p(\mathcal{Z}_{t'} = \mathbf{z}|\mathbf{z}_V))] \ .$$

(b) The work of Luna and Leonetti (2020) has defined the *relevance* of task t to task t' as the expected difference in the entropy of the policies before and after learning (over the states of t') w.r.t. the on-policy distribution before learning. For supervised meta-learning, such a relevance translates to

$$\rho(t, t') = H(\mathcal{Z}_{t'}|\mathcal{Z}_t) - H(\mathcal{Z}_t) \ .$$

Apart from these two components, the rest of the ITTS algorithm follows the original implementation.

Fig. 7 shows results of Sinusoid regression comparing the meta-test performance of MILT vs. ITTS. It can be observed that MILT outperforms ITTS for both 5-shot and 10-shot Sinusoid regression. Note that ITTS does

[16] To align with our problem setting where the remaining datasets for all tasks in A are acquired/observed for meta-training only after A is selected by an active task selection algorithm (Sections 3 and 2), only the *a priori* known sample datasets for all tasks in V are given during active task selection.

not outperform random selection when k is not sufficiently large. This may be due to its validation set (whose size $k' < k$) not being highly representative of the distribution of meta-test tasks. As a result, the calculation of the difference (see (a) above) using such a validation set can be misleading. When k is sufficiently large, the validation set becomes representative enough for ITTS to perform better than random selection.

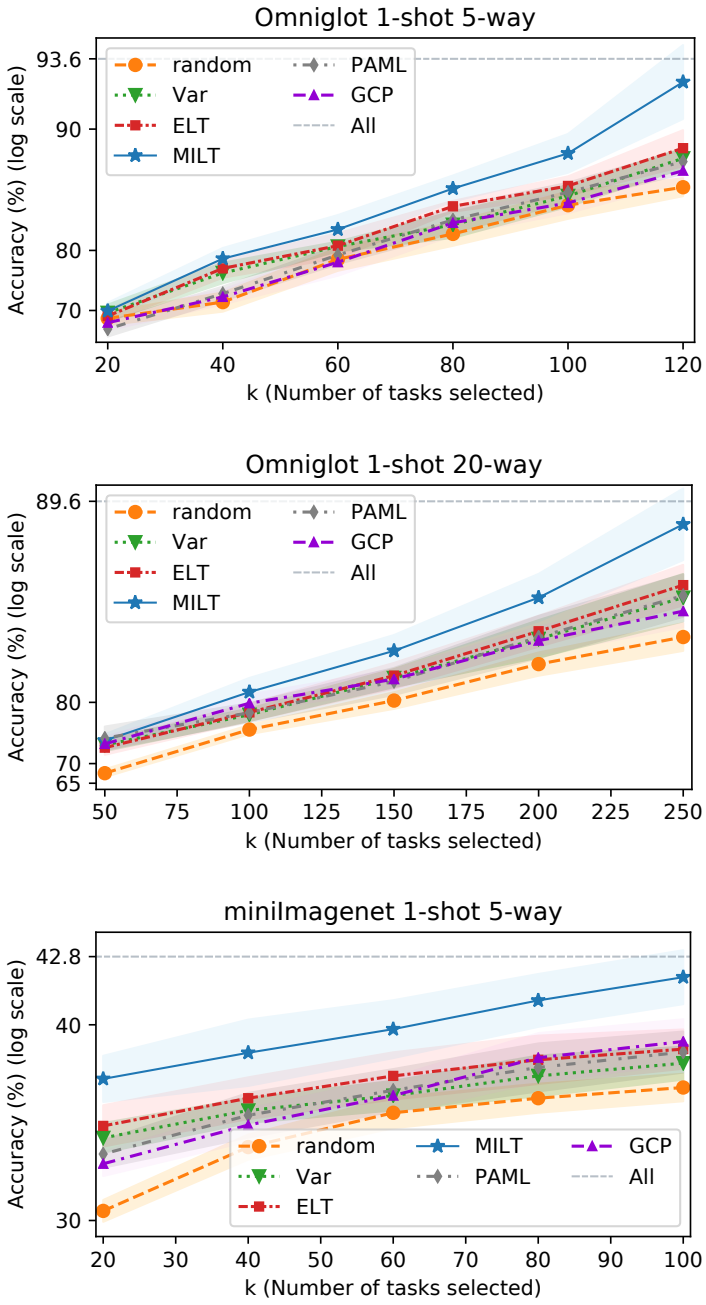

Figure 6: Meta-test accuracy and standard error over 5 runs vs. no. k of selected tasks on (top) 1-shot 5-way Omniglot, (middle) 1-shot 20-way Omniglot, and (bottom) 1-shot 5-way MiniImageNet.

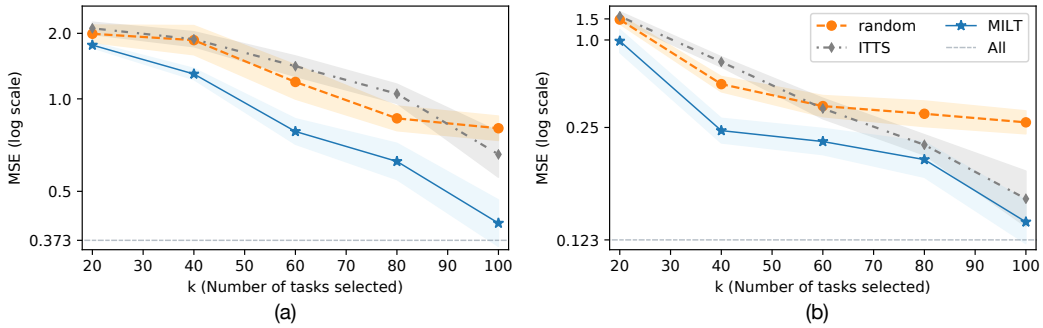

Figure 7: Meta-test *mean squared error* (MSE) and standard error over 5 runs vs. no. k of selected tasks on (a) 5-shot Sinusoid and (b) 10-shot Sinusoid.