Meta-learning Adaptive Deep Kernel Gaussian Processes for Molecular Property Prediction

Wenlin Chen
University of Cambridge
MPI for Intelligent Systems
wc337@cam.ac.uk

Austin Tripp
University of Cambridge
ajt212@cam.ac.uk

José Miguel Hernández-Lobato
University of Cambridge
jmh233@cam.ac.uk

Abstract

We propose Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT), a novel framework for learning deep kernel Gaussian processes (GPs) by interpolating between meta-learning and conventional deep kernel learning. Our approach employs a bilevel optimization objective where we meta-learn generally useful feature representations across tasks, in the sense that task-specific GP models estimated on top of such features achieve the lowest possible predictive loss on average. We solve the resulting nested optimization problem using the implicit function theorem (IFT). We show that our ADKF-IFT framework contains previously proposed Deep Kernel Learning (DKL) and Deep Kernel Transfer (DKT) as special cases. Although ADKF-IFT is a completely general method, we argue that it is especially well-suited for drug discovery problems and demonstrate that it significantly outperforms previous state-of-the-art methods on a variety of real-world few-shot molecular property prediction tasks and out-of-domain molecular property prediction and optimization tasks.

1 Introduction

Many real-world applications require machine learning algorithms to make robust predictions with well-calibrated uncertainty given very limited training data. One important example is drug discovery, where practitioners not only want models to accurately predict biochemical/physicochemical properties of molecules, but also want to use models to guide the search for novel molecules with desirable properties, leveraging techniques such as Bayesian optimization (BO) which heavily rely on accurate uncertainty estimates [11]. Despite the meteoric rise of neural networks over the past decade, their notoriously overconfident and unreliable uncertainty estimates [53] make them generally ineffective surrogate models for BO. Instead, most contemporary BO implementations use Gaussian processes (GPs) [45] as surrogate models due to their analytically-tractable and generally reliable uncertainty estimates, even on small datasets.

Traditionally, GPs are fit on hand-engineered features (e.g., molecular fingerprints), which can limit their predictive performance on complex, structured, high-dimensional data where designing informative features is challenging (e.g., molecules). Naturally, a number of works have proposed to improve performance by instead fitting GPs on features learned by a deep neural network: a family of models generally called Deep Kernel GPs. However, there is no clear consensus about how to train these models: maximizing the GP marginal likelihood [18, 63] has been shown to overfit on small datasets [37], while meta-learning [40] and fully-Bayesian approaches [37] avoid this at the cost of making strong, often unrealistic assumptions. This suggests that there is demand for new, better techniques for training deep kernel GPs.

In this work, we present a novel, general framework called *Adaptive Deep Kernel Fitting with Implicit Function Theorem* (ADKF-IFT) for training deep kernel GPs which we believe is especially well-suited to small datasets. ADKF-IFT essentially trains a subset of the model parameters with a

NeurIPS 2022 AI for Science Workshop.

meta-learning loss, and separately adapts the remaining parameters on each task using maximum marginal likelihood. In contrast to previous methods which use a single loss for all parameters, ADKF-IFT is able to utilize the implicit regularization of meta-learning to prevent overfitting while avoiding the strong assumptions of a pure meta-learning approach which may lead to underfitting. The key contributions and outline of the paper are as follows:

1. As our main technical contribution, we present the general ADKF-IFT framework and its natural formulation as a bilevel optimization problem (Section 3.1), then explain how the implicit function theorem (IFT) can be used to efficiently solve it with gradient-based methods in a few-shot learning setting (Section 3.2).
2. We show how ADKF-IFT can be viewed as a *generalization* and *unification* of previous approaches based purely on single-task learning [63] or purely on meta-learning [40] for training deep kernel GPs (Section 3.3).
3. We propose a specific practical instantiation of ADKF-IFT wherein all feature extractor parameters are meta-learned, which has a clear interpretation and obviates the need for any Hessian approximations. We argue why this particular instantiation is well-suited to retain the best properties of previously proposed methods (Section 3.4).
4. Motivated by the general demand for better GP models in chemistry, we perform an extensive empirical evaluation of ADKF-IFT on several chemical tasks, finding that it significantly improves upon previous state-of-the-art methods (Section 4).

2 Background and Notation

Gaussian Processes (GPs) are tools for specifying Bayesian priors over functions [45]. A $\mathcal{GP}(m_{\boldsymbol{\theta}}(\cdot), c_{\boldsymbol{\theta}}(\cdot,\cdot))$ is fully specified by a mean function $m_{\boldsymbol{\theta}}(\cdot)$ and a symmetric positive-definite covariance function $c_{\boldsymbol{\theta}}(\cdot,\cdot)$. The covariance function encodes the inductive bias (e.g., smoothness) of a GP. One advantage of GPs is that it is easy to perform principled model selection for its hyperparameters $\boldsymbol{\theta} \in \Theta$ using the marginal likelihood $p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})$ evaluated on the training data $(\mathbf{X}, \mathbf{y})$ and to obtain closed-form probabilistic predictions $p(\mathbf{y}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ for the test data $(\mathbf{X}_*, \mathbf{y}_*)$; we refer the readers to [45] for more details.

Deep Kernel Gaussian Processes are GPs whose covariance function is constructed by first using a neural network *feature extractor* $\mathbf{f}_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\phi} \in \Phi$ to create feature representations $\mathbf{h} = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{h}' = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}')$ of the input points $\mathbf{x}, \mathbf{x}'$, then feeding these feature representations into a standard *base kernel* $c_{\boldsymbol{\theta}}(\mathbf{h}, \mathbf{h}')$ (e.g., an RBF kernel) [18, 63, 62, 3, 4]. The complete covariance function is therefore $k_{\boldsymbol{\psi}}(\mathbf{x}, \mathbf{x}') = c_{\boldsymbol{\theta}}(\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}'))$ with learnable parameters $\boldsymbol{\psi} = (\boldsymbol{\theta}, \boldsymbol{\phi})$.

Few-shot Learning refers to learning on many related tasks when each task has few labelled examples [35, 26]. In the standard problem setup, one is given a set of training tasks $\mathcal{D} = \{\mathcal{T}_t\}_{t=1}^{T}$ (a *meta-dataset*) and some unseen test tasks $\mathcal{D}_* = \{\mathcal{T}_*\}$. Each task $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{\mathcal{T}}}$ is a set of points in the domain $\mathcal{X}$ (e.g., space of molecules) with corresponding labels (continuous, categorical, etc.), and is partitioned into a *support set* $\mathcal{S}_{\mathcal{T}} \subseteq \mathcal{T}$ for training and a *query set* $\mathcal{Q}_{\mathcal{T}} = \mathcal{T} \setminus \mathcal{S}_{\mathcal{T}}$ for testing. Typically, the total number of training tasks $T = |\,\mathcal{D}\,|$ is large, while the size of each support set $|\,\mathcal{S}_{\mathcal{T}}\,|$ is small. Models for few-shot learning are typically trained to accurately predict $\mathcal{Q}_{\mathcal{T}}$ given $\mathcal{S}_{\mathcal{T}}$ for $\mathcal{T} \in \mathcal{D}$ during a *meta-training* phase, then evaluated by their prediction error on $\mathcal{Q}_{\mathcal{T}_*}$ given $\mathcal{S}_{\mathcal{T}_*}$ for unseen test tasks $\mathcal{T}_* \in \mathcal{D}_*$ during a *meta-testing* phase.

3 Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT)

3.1 The General ADKF-IFT Framework for Learning Deep Kernel GPs

Let A_Θ and A_Φ respectively be the sets of base kernel and feature extractor parameters for a deep kernel GP. Denote the set of all parameters by $A_\Psi = A_\Theta \cup A_\Phi$. The key idea of the general ADKF-IFT framework is that only a subset of the parameters $A_{\Psi_{\text{adapt}}} \subseteq A_\Psi$ will be adapted to each individual task by minimizing a train loss $\mathcal{L}_T$, with the remaining set of parameters $A_{\Psi_{\text{meta}}} = A_\Psi \setminus A_{\Psi_{\text{adapt}}}$ meta-learned during a *meta-training* phase to yield the best possible validation loss $\mathcal{L}_V$ *on average* over many related training tasks (*after* $A_{\Psi_{\text{adapt}}}$ is separately adapted to each of these tasks). This can

be naturally formalized as the following *bilevel optimization* problem:

$$\boldsymbol{\psi}^*_{\text{meta}} = \arg\min_{\boldsymbol{\psi}_{\text{meta}}} \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})], \quad (1)$$

$$\text{such that} \quad \boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \arg\min_{\boldsymbol{\psi}_{\text{adapt}}} \mathcal{L}_T(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{S}_\mathcal{T}). \quad (2)$$

Equations (1) and (2) are most easily understood by separately considering the meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$ and the task-specific parameters $\boldsymbol{\psi}_{\text{adapt}}$. For a given task $\mathcal{T}$ and an arbitrary value for the meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$, in Equation (2) the task-specific parameters $\boldsymbol{\psi}_{\text{adapt}}$ are chosen to minimize the *train loss* $\mathcal{L}_T$ evaluated on the task's support set $\mathcal{S}_\mathcal{T}$. That is, $\boldsymbol{\psi}_{\text{adapt}}$ is *adapted* to the support set $\mathcal{S}_\mathcal{T}$ of the task $\mathcal{T}$, with the aim of producing the best possible model on $\mathcal{S}_\mathcal{T}$ for the given value of $\boldsymbol{\psi}_{\text{meta}}$. The result is a model with optimal task-specific parameters $\boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}_{\text{meta}}, \mathcal{S}_\mathcal{T})$ for the given meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$ and task $\mathcal{T}$. The remaining question is how to choose a value for the meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$, knowing that $\boldsymbol{\psi}_{\text{adapt}}$ will be adapted separately to each task. In Equation (1), we propose to choose $\boldsymbol{\psi}_{\text{meta}}$ to minimize the expected *validation loss* $\mathcal{L}_V$ over a distribution of training tasks $p(\mathcal{T})$. There are two reasons for this. First, on any given task $\mathcal{T}$, the validation loss usually reflects the performance metric of interest on the query set $\mathcal{Q}_\mathcal{T}$ of $\mathcal{T}$ (e.g., the prediction error). Second, because the same value of $\boldsymbol{\psi}_{\text{meta}}$ will be used for all tasks, it makes sense to choose a value whose *expected performance* is good across many tasks drawn from $p(\mathcal{T})$. That is, $\boldsymbol{\psi}_{\text{meta}}$ is chosen such that a GP achieves the lowest possible average validation loss on the query set $\mathcal{Q}_\mathcal{T}$ of a random training task $\mathcal{T} \sim p(\mathcal{T})$ after $\boldsymbol{\psi}_{\text{adapt}}$ is adapted to the task's support set $\mathcal{S}_\mathcal{T}$.

In practice, $\boldsymbol{\psi}_{\text{meta}}$ would be optimized during a *meta-training* phase using a set of training tasks $\mathcal{D}$ to approximate Equation (1). After meta-training (i.e., at meta-test time), we make predictions for each unseen test task $\mathcal{T}_*$ using the joint GP posterior predictive distribution with optimal parameters $\boldsymbol{\psi}^*_{\text{meta}}$ and $\boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}^*_{\text{meta}}, \mathcal{S}_{\mathcal{T}_*})$:

$$p(\mathcal{Q}^y_{\mathcal{T}_*} \mid \mathcal{Q}^{\mathbf{x}}_{\mathcal{T}_*}, \mathcal{S}_{\mathcal{T}_*}, \boldsymbol{\psi}^*_{\text{meta}}, \boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}^*_{\text{meta}}, \mathcal{S}_{\mathcal{T}_*})). \quad (3)$$

Note that the description above does not specify a particular choice of $A_{\Psi_{\text{meta}}}, A_{\Psi_{\text{adapt}}}, \mathcal{L}_T, \mathcal{L}_V$. This is intentional, as there are many reasonable choices for these quantities. Because of this, we believe that ADKF-IFT should be considered a *general framework*, with a particular choice for these being an *instantiaton* of the ADKF-IFT framework. We give examples of this in Sections 3.3 and 3.4.

3.2 Efficient Meta-Training Algorithm

In general, optimizing bilevel optimization objectives such as Equation (1) is computationally complex, mainly because each evaluation of the objective requires solving a separate inner optimization problem (2). Although calculating the *hypergradient* (i.e., total derivative) of the validation loss $\mathcal{L}_V$ w.r.t. the meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$ would allow Equation (1) to be solved with gradient-based optimization:

$$\frac{d\,\mathcal{L}_V}{d\,\boldsymbol{\psi}_{\text{meta}}} = \frac{\partial\,\mathcal{L}_V}{\partial\,\boldsymbol{\psi}_{\text{meta}}} + \frac{\partial\,\mathcal{L}_V}{\partial\,\boldsymbol{\psi}^*_{\text{adapt}}} \frac{\partial\,\boldsymbol{\psi}^*_{\text{adapt}}}{\partial\,\boldsymbol{\psi}_{\text{meta}}}, \quad (4)$$

Equation (4) reveals that this requires calculating $\partial\,\boldsymbol{\psi}^*_{\text{adapt}} / \partial\,\boldsymbol{\psi}_{\text{meta}}$, i.e., how the optimal task-specific parameters $\boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}_{\text{meta}}, \mathcal{S}_\mathcal{T})$ change with respect to the meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$. Calculating this naively with automatic differentiation platforms would require tracking the gradient through many iterations of the inner optimization (2), which in practice requires too much memory to be feasible. Fortunately, because $\boldsymbol{\psi}^*_{\text{adapt}}$ is an optimum of the train loss $\mathcal{L}_T$, Cauchy's *Implicit Function Theorem* (IFT) provides a formula for calculating $\partial\,\boldsymbol{\psi}^*_{\text{adapt}} / \partial\,\boldsymbol{\psi}_{\text{meta}}$ for an arbitrary value of the meta-learned parameters $\boldsymbol{\psi}'_{\text{meta}}$ and a given task $\mathcal{T}'$:

$$\left.\frac{\partial\,\boldsymbol{\psi}^*_{\text{adapt}}}{\partial\,\boldsymbol{\psi}_{\text{meta}}}\right|_{\boldsymbol{\psi}'_{\text{meta}}} = -\left(\frac{\partial^2\,\mathcal{L}_T(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial\,\boldsymbol{\psi}_{\text{adapt}}\,\partial\,\boldsymbol{\psi}^T_{\text{adapt}}}\right)^{-1} \left.\frac{\partial^2\,\mathcal{L}_T(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial\,\boldsymbol{\psi}_{\text{adapt}}\,\partial\,\boldsymbol{\psi}^T_{\text{meta}}}\right|_{\boldsymbol{\psi}'_{\text{meta}}, \boldsymbol{\psi}'_{\text{adapt}}}, \quad (5)$$

where $\boldsymbol{\psi}'_{\text{adapt}} = \boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$. A full statement of the implicit function theorem in the context of ADKF-IFT can be found in Appendix A. The only potential problem with Equation (5) is the computation and inversion of the Hessian matrix $\partial^2\,\mathcal{L}_T(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{S}_\mathcal{T}) / \partial\,\boldsymbol{\psi}_{\text{adapt}}\,\partial\,\boldsymbol{\psi}^T_{\text{adapt}}$. This computation can be done exactly if $|A_{\Psi_{\text{adapt}}}|$ is small, which is the case considered in this paper (as will be discussed

Algorithm 1 Exact hypergradient computation in ADKF-IFT.

1: **Input:** a training task $\mathcal{T}'$ and the current meta-learned parameters $\boldsymbol{\psi}'_{\text{meta}}$.
2: Solve Equation (2) to obtain $\boldsymbol{\psi}'_{\text{adapt}} = \boldsymbol{\psi}^*_{\text{adapt}}(\boldsymbol{\psi}'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$.
3: Compute $\mathbf{g}_1 = \left.\frac{\partial \mathcal{L}_V(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{T}')}{\partial \boldsymbol{\psi}_{\text{meta}}}\right|_{\boldsymbol{\psi}'_{\text{meta}}, \boldsymbol{\psi}'_{\text{adapt}}}$ and $\mathbf{g}_2 = \left.\frac{\partial \mathcal{L}_V(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{T}')}{\partial \boldsymbol{\psi}_{\text{adapt}}}\right|_{\boldsymbol{\psi}'_{\text{meta}}, \boldsymbol{\psi}'_{\text{adapt}}}$ by auto-diff.
4: Compute the Hessian $\mathbf{H} = \left.\frac{\partial^2 \mathcal{L}_T(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{\text{adapt}} \partial \boldsymbol{\psi}_{\text{adapt}}^T}\right|_{\boldsymbol{\psi}'_{\text{meta}}, \boldsymbol{\psi}'_{\text{adapt}}}$ by auto-diff.
5: Solve the linear system $\mathbf{v}\,\mathbf{H} = \mathbf{g}_2$ for $\mathbf{v}$.
6: Compute the mixed partial derivatives $\mathbf{P} = \left.\frac{\partial^2 \mathcal{L}_T(\boldsymbol{\psi}_{\text{meta}}, \boldsymbol{\psi}_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{\text{adapt}} \partial \boldsymbol{\psi}_{\text{meta}}^T}\right|_{\boldsymbol{\psi}'_{\text{meta}}, \boldsymbol{\psi}'_{\text{adapt}}}$ by auto-diff.
7: **Output:** the hypergradient $\frac{d\,\mathcal{L}_V}{d\,\boldsymbol{\psi}_{\text{meta}}} = \mathbf{g}_1 - \mathbf{v}\,\mathbf{P}$. ▷ Equations (4) and (5)

in Section 3.4). Otherwise, an approximation to the inverse Hessian (e.g., Neumann approximation [31, 7]) could be used, which reduces both the memory and computational complexities to $\mathcal{O}(|A_\Psi|)$. Combining Equations (4) and (5), we have a recipe for computing the hypergradient $d\,\mathcal{L}_V / d\,\boldsymbol{\psi}_{\text{meta}}$ exactly for a single task, as summarized in Algorithm 1. The meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$ can then be updated with the expected hypergradient over $p(\mathcal{T})$.

3.3 ADKF-IFT as a Unification of Previous Methods

In prior work, the most common method used to train deep kernel GPs is to minimize the negative log marginal likelihood (NLML) on a single dataset (optionally with extra regularization terms). This is commonly referred to as *Deep Kernel Learning* (DKL) [63], and is stated explicitly in Equation (6). The most notable departure from DKL is *Deep Kernel Transfer* (DKT) [40], which instead proposes to train deep kernel GPs entirely using meta-learning, minimizing the *expected* NLML over a distribution of training tasks, as is stated explicitly in Equation (7).

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \text{NLML}\left(\boldsymbol{\psi}, \mathcal{S}_{\mathcal{T}}\right) \quad (6) \qquad \boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \mathbb{E}_{p(\mathcal{T})}[\text{NLML}(\boldsymbol{\psi}, \mathcal{T})] \quad (7)$$

Interestingly, both DKL and DKT can be viewed as *special cases* of the general ADKF-IFT framework. It is simple to see that choosing the partition to be $A_{\Psi_{\text{meta}}} = \varnothing$, $A_{\Psi_{\text{adapt}}} = A_\Psi$ and the train loss $\mathcal{L}_T$ to be the NLML in Equations (1) and (2) yields Equation (6): DKL is just ADKF-IFT if no parameters are meta-learned. Similarly, choosing the partition to be $A_{\Psi_{\text{meta}}} = A_\Psi$, $A_{\Psi_{\text{adapt}}} = \varnothing$ and the validation loss $\mathcal{L}_V$ to be the NLML in Equations (1) and (2) yields Equation (7): DKT is just ADKF-IFT if all parameters are meta-learned. This makes ADKF-IFT *strictly more general* than these two methods.

3.4 Highlighted ADKF-IFT Instantiation: Meta-learn $\boldsymbol{\phi}$, Adapt $\boldsymbol{\theta}$

Among the many possible variations of ADKF-IFT, we wish to highlight the following instantation:

- The train loss $\mathcal{L}_T$ and validation loss $\mathcal{L}_V$ are the negative log GP marginal likelihood on $\mathcal{S}_{\mathcal{T}}$ and the negative log joint GP predictive posterior on $\mathcal{Q}_{\mathcal{T}}$, respectively.
- $A_{\Psi_{\text{meta}}} = A_\Phi$, i.e., all feature extractor parameters $\boldsymbol{\phi}$ are meta-learned.
- $A_{\Psi_{\text{adapt}}} = A_\Theta$, i.e., all base kernel parameters $\boldsymbol{\theta}$ are adapted (e.g., noise, lengthscales).

There are several benefits to this choice. First, this particular choice of loss functions has the advantage that the prediction procedure during meta-testing (as defined in Equation (3)) exactly matches the meta-training procedure, thereby closely following the principle of *learning to learn*. Second, the partition of parameters can be intuitively understood as meta-learning a generally useful feature extractor $\mathbf{f}_{\boldsymbol{\phi}}$ such that it is possible on average to fit a low-loss GP to the feature representations extracted by $\mathbf{f}_{\boldsymbol{\phi}}$ for each individual task. This is very similar to previous transfer learning approaches. Third, since most GP base kernels have only a handful of parameters, the Hessian in Equation (5) can be computed and inverted exactly during meta-training using Algorithm 1; this removes any need for Hessian approximations. Fourth, the inner optimization (2) for $\boldsymbol{\psi}_{\text{adapt}}$ is computationally efficient, as it does not require backpropagating through the feature extractor $\mathbf{f}_{\boldsymbol{\phi}}$.

More generally, we conjecture that adapting just the base kernel parameters will allow ADKF-IFT to achieve a better balance between overfitting and underfitting than either DKL or DKT. The

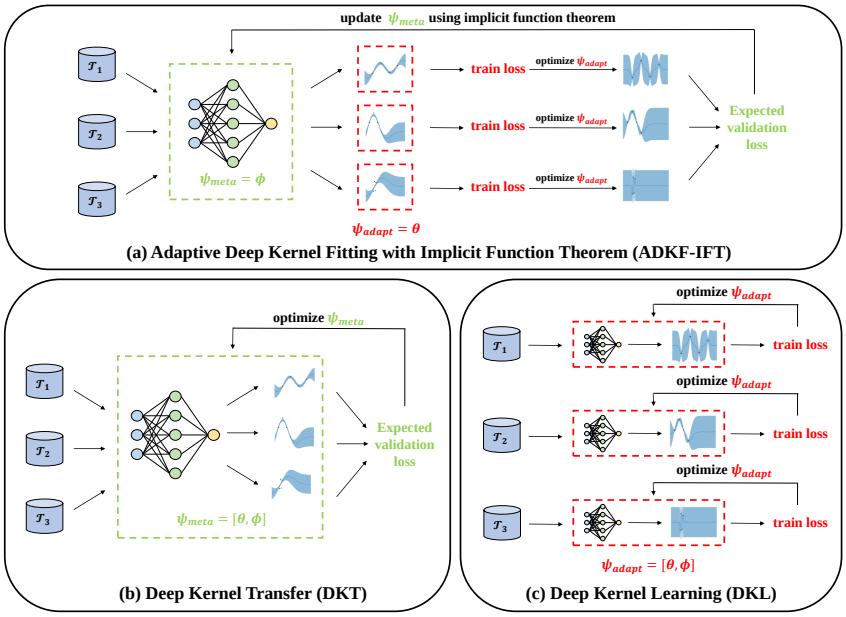


Figure 1: A contrastive diagram illustrating the training procedures of ADKF-IFT, DKT, and DKL.

relationship between these methods are visualized in Figure 1. Panel (c) shows DKL, which trains a separate deep kernel GP for each task. It is not hard to imagine that this can lead to severe overfitting for small datasets, which has been observed empirically by [37]. Panel (b) shows DKT, which prevents overfitting by fitting one deep kernel GP for all tasks. However, this implicitly makes a strong assumption that all tasks come from an identical distribution over functions, including the same noise level, same amplitude, and same characteristic lengthscales, which is unlikely to hold in practice. Panel (a) shows ADKF-IFT, which allows these important parameters to be adapted, while still regularizing the feature extractor with meta-learning. We conjecture that adapting the base kernel parameters is more appropriate given the expected differences between tasks: two related tasks are more likely to have different noise levels or characteristic lengthscales than to require substantially different feature representations. We refer the readers to Appendix I for more discussions.

4 Experiments

In this section, we evaluate the empirical performance of ADKF-IFT from Section 3.4. We choose to focus our experiments exclusively on molecular property prediction and optimization tasks because we believe this application would benefit greatly from better GP models: firstly because many existing methods struggle on small datasets of size $\sim 10^2$ which are ubiquitous in chemistry, and secondly because many tasks in chemistry require high-quality uncertainty estimates. First, we evaluate ADKF-IFT on four commonly used benchmark tasks from MoleculeNet [64], finding that ADKF-IFT achieves state-of-the-art results on most tasks (Section 4.1). Second, we evaluate ADKF-IFT on the larger-scale FS-Mol benchmark [50], finding that ADKF-IFT is the best-performing method (Section 4.2). In particular, our results support the hypothesis from Section 3.4 that ADKF-IFT achieves a better balance between overfitting and underfitting than DKL and DKT. Finally, we show that the ADKF-IFT feature representation is transferable to out-of-domain molecular property prediction and optimization tasks (Section 4.3). The general configurations of ADKF-IFT for all experiments considered in this paper are shown in Appendix C.

4.1 Few-shot Molecular Property Prediction on the MoleculeNet Benchmark

Benchmark and Baselines. We compare **ADKF-IFT** with two types of baselines on four few-shot molecular property classification benchmark tasks (Tox21, SIDER, MUV, ToxCast) from MoleculeNet [64] (see Appendix D for more details of MoleculeNet benchmark tasks): 1) methods with feature extractor trained from scratch: **Siamese** [25], **ProtoNet** [48], **MAML** [10], **TPN** [30], **EGNN** [22], **IterRefLSTM** [1] and **PAR** [59]; and 2) methods that fine-tune a pretrained feature extractor: **Pre-**

Table 1: Mean test performance (AUROC%) with standard deviations of all compared methods on MoleculeNet benchmark tasks at support set size 20 (i.e., 2-way 10-shot).

Method	MoleculeNet benchmark task (#compounds in total)			
	Tox21 (8,014)	SIDER (1,427)	MUV (93,127)	ToxCast (8,615)
Siamese	80.40 ± 0.35	71.10 ± 4.32	59.59 ± 5.13	-
ProtoNet	74.98 ± 0.32	64.54 ± 0.89	65.88 ± 4.11	63.70 ± 1.26
MAML	80.21 ± 0.24	70.43 ± 0.76	63.90 ± 2.28	66.79 ± 0.85
TPN	76.05 ± 0.24	67.84 ± 0.95	65.22 ± 5.82	62.74 ± 1.45
EGNN	81.21 ± 0.16	72.87 ± 0.73	65.20 ± 2.08	63.65 ± 1.57
IterRefLSTM	81.10 ± 0.17	69.63 ± 0.31	45.56 ± 5.12	-
PAR	82.06 ± 0.12	$\mathbf{74.68 \pm 0.31}$	66.48 ± 2.12	69.72 ± 1.63
ADKF-IFT	$\mathbf{82.43 \pm 0.60}$	67.72 ± 1.21	$\mathbf{98.18 \pm 3.05}$	$\mathbf{72.07 \pm 0.81}$
Pre-GNN	82.14 ± 0.08	73.96 ± 0.08	67.14 ± 1.58	73.68 ± 0.74
Meta-MGNN	82.97 ± 0.10	75.43 ± 0.21	68.99 ± 1.84	-
Pre-PAR	84.93 ± 0.11	$\mathbf{78.08 \pm 0.16}$	69.96 ± 1.37	75.12 ± 0.84
Pre-ADKF-IFT	$\mathbf{86.06 \pm 0.35}$	70.95 ± 0.60	$\mathbf{95.74 \pm 0.37}$	$\mathbf{76.22 \pm 0.13}$

GNN [19], **Meta-MGNN** [16] and **Pre-PAR** [59]. **Pre-ADKF-IFT** refers to ADKF-IFT starting from a pretrained feature extractor. All compared methods in this section use GIN [65] as their feature extractors. The pretrained weights for the methods of the second type are provided by [19].

Evaluation Procedure. We follow exactly the same evaluation procedure as that in [59, 19, 16]. The task-level metric is AUROC (area under the receiver operating characteristic curve). We report the averaged performance over ten runs with different random seeds for each compared method at the support set size 20 (i.e., 2-way 10-shot, as the support sets in MoleculeNet are balanced). We did not perform 1-shot learning, as it is an unrealistic setting in real-world drug discovery tasks. All baseline results are taken from [59].

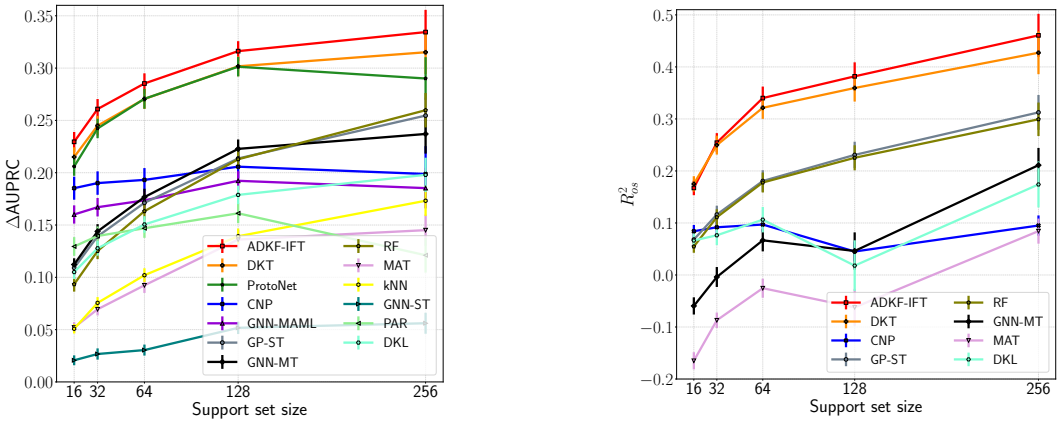
Performance. Table 1 shows that ADKF-IFT and Pre-ADKF-IFT achieve the best performance on Tox21, MUV, and ToxCast. In general, the larger the dataset is, the larger the performance gains of our method over other baselines are, highlighting the scalability of our method. In particular, our method outperforms all baselines by a wide margin on MUV due to the relatively large amount of available compounds, but underperforms many baselines on SIDER due to a lack of compounds.

4.2 Few-shot Molecular Property Prediction on the FS-Mol Benchmark

Benchmark. We further conduct our evaluation on the FS-Mol benchmark [50], which contains a carefully constructed set of few-shot learning tasks for molecular property prediction. FS-Mol contains over 5,000 tasks with 233,786 unique compounds from ChEMBL27 [34], split into training (4,938 tasks), validation (40 tasks), and test (157 tasks) sets. Each task is associated with a protein target. The original benchmark only considers binary classification of active/inactive compounds, but we include the regression task (for the actual numeric activity target IC50 or EC50) in our evaluation as well, as it is a desired and more preferred task to do in real-world drug discovery projects.

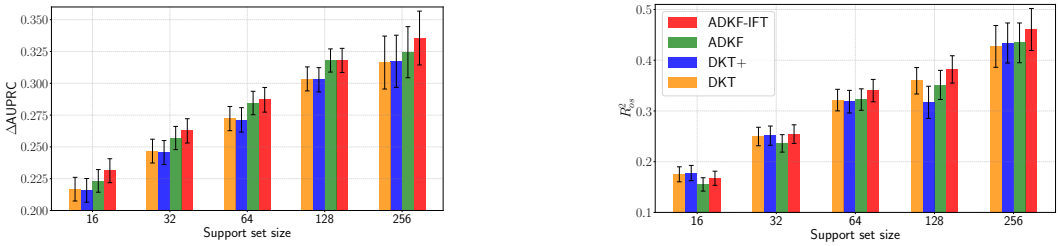
Baselines. We compare **ADKF-IFT** with four categories of baselines: 1) single-task methods: Random Forest (**RF**), k-Nearest Neighbors (**kNN**), single-task GP with Tanimoto kernel (**GP-ST**) [44], single-task GNN (**GNN-ST**) [15], Deep Kernel Learning (**DKL**) [63]; 2) multi-task pretraining: multi-task GNN (**GNN-MT**) [9, 15]; 3) self-supervised pretraining: Molecule Attention Transformer (**MAT**) [33]; 4) meta-learning methods: Property-Aware Relation Networks (**PAR**) [59], Prototypical Network with Mahalanobis distance (**ProtoNet**) [48], Model-Agnostic Meta-Learning (**GNN-MAML**) [10], Conditional Neural Process (**CNP**) [13], Deep Kernel Transfer (**DKT**) [40]. The GNN feature extractor architecture $\mathbf{f}_{\phi}$ used for DKL, PAR, CNP, DKT, and ADKF-IFT is the same as that used for ProtoNet, GNN-ST, GNN-MT, and GNN-MAML in [50]. All multi-task and meta-learning methods are trained from scratch on FS-Mol training tasks. MAT is pretrained on 2 millions molecules sampled from the ZINC15 dataset [51]. The classification results for RF, kNN, GNN-ST, GNN-MT, MAT, ProtoNet, and GNN-MAML are reproduced according to [50]. Detailed configurations of all compared methods can be found in Appendix E.

Evaluation Procedure. The task-level metrics for binary classification and regression are ΔAUPRC (change in area under the precision-recall curve) and R^2_{os} (predictive/out-of-sample coefficient of determination), respectively. Details of these metrics can be found in Appendix F. We follow exactly the same evaluation procedure as that in [50], where the averaged performance over ten different stratified support/query random splits of every test task is reported for each compared method. This



(a) Classification (157 tasks). (b) Regression (111 tasks).

Figure 2: Mean performance with standard errors of all compared methods on all FS-Mol test tasks.



(a) Classification (157 tasks). (b) Regression (111 tasks).

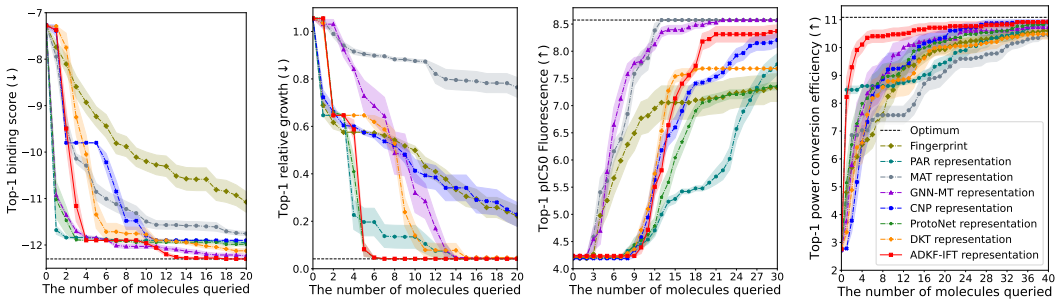
Figure 3: Mean performance with standard errors of ablation models on all FS-Mol test tasks. ADKF is like ADKF-IFT but assuming $\partial \boldsymbol{\theta}^*/\partial \boldsymbol{\phi} = \mathbf{0}$, i.e., updating $\boldsymbol{\phi}$ with the direct gradient $\partial \mathcal{L}_V/\partial \boldsymbol{\phi}$. DKT+ is like DKT but tuning the base kernel parameters $\boldsymbol{\theta}$ during meta-testing.

evaluation process is performed for five different support set sizes 16, 32, 64, 128, and 256. Note that the support sets are generally unbalanced for the classification task in FS-Mol, which is natural as the majority of the candidate molecules are inactive in drug discovery.

Overall Performance. Figure 2 shows the overall test performance of all compared methods. Note that RF is a strong baseline method, as it is widely used in real-world drug discovery projects and has comparable performance to many pretraining methods. The results indicate that ADKF-IFT outperforms all the other compared methods at all considered support set sizes for the classification task. For the regression task, the performance gains of ADKF-IFT over the second best method, namely DKT, get larger as the support set size increases. In Appendix G.1, we show that ADKF-IFT achieves the best mean rank for both classification and regression at all considered support set sizes.

Statistical Comparison. We perform two-sided Wilcoxon signed-rank tests [61] to compare the performance of ADKF-IFT and the next best method, namely DKT. The exact p-values from these statistical tests can be found in Appendix G.2. The results indicate that ADKF-IFT *significantly* outperforms DKT for the classification task at all considered support set sizes and for the regression task at support set sizes 64, 128, and 256 (at significance level $\alpha = 0.05$).

Ablation Study. To show that 1) the bilevel optimization objective for ADKF-IFT is essential for learning informative feature representations and 2) the performance gains of ADKF-IFT are not simply caused by tuning the base kernel parameters $\boldsymbol{\theta}$ at meta-test time, we consider two ablation models: DKT+ and ADKF. The test performance of these models are shown in Figure 3. For ADKF, we follow the ADKF-IFT training scheme but assume $\partial \boldsymbol{\theta}^*/\partial \boldsymbol{\phi} = \mathbf{0}$, i.e., updating the feature extractor parameters $\boldsymbol{\phi}$ with the *direct gradient* $\partial \mathcal{L}_V/\partial \boldsymbol{\phi}$ rather than $d \mathcal{L}_V/d \boldsymbol{\phi}$. The results show that ADKF consistently underperforms ADKF-IFT, indicating that the hypergradient for the bilevel optimization objective has non-negligible contributions to learning better feature representations. For DKT+, we take a model trained by DKT and adapt the base kernel parameters $\boldsymbol{\theta}$ on each task at meta-test time.



(a) Molecular docking. (b) Antibiotic discovery. (c) Antiviral drug design. (d) Material design.

Figure 4: Mean top-1 target values with standard errors as a function of the number of molecules queried for all compared feature representations on four out-of-domain molecular optimization tasks.

Table 2: Mean predictive performance (test NLL) with standard errors of a GP operating on top of each compared feature representation on the four out-of-domain molecular design tasks.

Feature representation	Out-of-domain molecular design task			
	Molecular docking	Antibiotic discovery	Antiviral drug design	Material design
Fingerprint	1.138 ± 0.014	1.669 ± 0.075	$\mathbf{4.601 \pm 0.086}$	1.091 ± 0.011
PAR	1.270 ± 0.019	2.185 ± 0.115	4.840 ± 0.086	1.283 ± 0.017
MAT	1.528 ± 0.028	2.390 ± 0.104	4.797 ± 0.088	2.198 ± 0.063
GNN-MT	1.994 ± 0.050	3.692 ± 0.225	6.399 ± 0.181	7.254 ± 0.217
CNP	1.493 ± 0.028	2.537 ± 0.162	5.005 ± 0.086	1.741 ± 0.043
ProtoNet	1.147 ± 0.013	1.615 ± 0.094	5.060 ± 0.086	1.032 ± 0.009
DKT	1.167 ± 0.012	1.602 ± 0.073	4.975 ± 0.092	1.026 ± 0.009
ADKF-IFT	$\mathbf{1.137 \pm 0.011}$	$\mathbf{1.496 \pm 0.043}$	4.781 ± 0.087	$\mathbf{0.996 \pm 0.007}$

The results show that DKT+ does not improve upon DKT, indicating that tuning the base kernel parameters $\boldsymbol{\theta}$ at meta-test time is not sufficient for obtaining better test performance with DKT.

Sub-benchmark Performance. The tasks in FS-Mol can be partitioned into 7 sub-benchmarks by Enzyme Commission number [60]. In Appendix G.3, we show the test performance of top performing methods on each sub-benchmark. The results indicate that, in addition to achieving best overall performance, ADKF-IFT achieves the best performance on all sub-benchmarks for the regression task and on more than half of the sub-benchmarks for the classification task.

4.3 Out-of-domain Molecular Property Prediction and Optimization

Finally, we demonstrate that the feature representation learned by ADKF-IFT is useful not only for in-domain molecular property prediction tasks but also for out-of-domain molecular property prediction and optimization tasks. For this, we perform experiments involving finding molecules with best desired target properties within given out-of-domain datasets using Bayesian optimization (BO) with a GP surrogate model operating on top of compared feature representations. We use the expected improvement acquisition function [21] with query-batch size 1. All compared feature representations are extracted using the models trained on the FS-Mol dataset from scratch in Section 4.2, except for the pretrained MAT representation and fingerprint. We compare them on four representative molecular design tasks outside of FS-Mol. Detailed configuration of the GP and descriptions of the tasks can be found in Appendix H. We repeat each BO experiment 20 times, each time starting from 16 randomly sampled molecules from the worst ~ 700 molecules within the dataset. Figure 4 shows that the ADKF-IFT representation enables fastest discovery of top performing molecules for the molecular docking, antibiotic discovery, and material design tasks. For the antiviral drug design task, although the ADKF-IFT representation underperforms the MAT and GNN-MT representations, it still achieves competitive performance compared to other baselines.

Table 2 explicitly reports the regression predictive performance of a GP operating on top of each compared feature representation for these four out-of-domain molecular design tasks. The configuration of the GP is the same as that in the BO experiments. We report test negative log likelihood (NLL) averaged over 200 support/query random splits (100 for each of the support set sizes 32 and 64). The results show that the ADKF-IFT representation has the best test NLL on the molecular docking, antibiotic discovery, and material design tasks, and ranks second on the antiviral drug design task.

References

[1] Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, and Vijay S. Pande. Low data drug discovery with one-shot learning. *ACS Central Science*, 3:283 – 293, 2017.

[2] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[3] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

[4] Roberto Calandra, Jan Peters, Carl E Rasmussen, and Marc Peter Deisenroth. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016.

[5] Da Chen, Yuefeng Chen, Yuhong Li, Feng Mao, Yuan He, and Hui Xue. Self-supervised learning for few-shot image classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1745–1749. IEEE, 2021.

[6] Yutian Chen, Abram L Friesen, Feryal Behbahani, Arnaud Doucet, David Budden, Matthew Hoffman, and Nando de Freitas. Modular meta-learning with shrinkage. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2858–2869. Curran Associates, Inc., 2020.

[7] Ross M Clarke, Elre Talea Oldewage, and José Miguel Hernández-Lobato. Scalable one-pass optimisation of high-dimensional weight-update hyperparameters by implicit differentiation. In *International Conference on Learning Representations*, 2022.

[8] The COVID Moonshot Consortium, Hagit Achdout, Anthony Aimon, Elad Bar-David, Haim Barr, Amir Ben-Shmuel, James Bennett, Vitaliy A. Bilenko, Vitaliy A. Bilenko, Melissa L. Boby, Bruce Borden, Gregory R. Bowman, Juliane Brun, Sarma BVNBS, Mark Calmiano, Anna Carbery, Daniel Carney, Emma Cattermole, Edcon Chang, Eugene Chernyshenko, John D. Chodera, Austin Clyde, Joseph E. Coffland, Galit Cohen, Jason Cole, Alessandro Contini, Lisa Cox, Milan Cvitkovic, Alex Dias, Kim Donckers, David L. Dotson, Alice Douangamath, Shirly Duberstein, Tim Dudgeon, Louise Dunnett, Peter K. Eastman, Noam Erez, Charles J. Eyermann, Mike Fairhead, Gwen Fate, Daren Fearon, Oleg Fedorov, Matteo Ferla, Rafaela S. Fernandes, Lori Ferrins, Richard Foster, Holly Foster, Ronen Gabizon, Adolfo Garcia-Sastre, Victor O. Gawriljuk, Paul Gehrtz, Carina Gileadi, Charline Giroud, William G. Glass, Robert Glen, Itai Glinert, Andre S. Godoy, Marian Gorichko, Tyler Gorrie-Stone, Ed J. Griffen, Storm Hassell Hart, Jag Heer, Michael Henry, Michelle Hill, Sam Horrell, Victor D. Huliak, Matthew F.D. Hurley, Tomer Israely, Andrew Jajack, Jitske Jansen, Eric Jnoff, Dirk Jochmans, Tobias John, Steven De Jonghe, Anastassia L. Kantsadi, Peter W. Kenny, J. L. Kiappes, Serhii O. Kinakh, Lizbe Koekemoer, Boris Kovar, Tobias Krojer, Alpha Lee, Bruce A. Lefker, Haim Levy, Ivan G. Logvinenko, Nir London, Petra Lukacik, Hannah Bruce Macdonald, Beth MacLean, Tika R. Malla, Tatiana Matviiuk, Willam McCorkindale, Briana L. McGovern, Sharon Melamed, Kostiantyn P. Melnykov, Oleg Michurin, Halina Mikolajek, Bruce F. Milne, Aaron Morris, Garrett M. Morris, Melody Jane Morwitzer, Demetri Moustakas, Aline M. Nakamura, Jose Brandao Neto, Johan Neyts, Luong Nguyen, Gabriela D. Noske, Vladas Oleinikovas, Glaucius Oliva, Gijs J. Overheul, David Owen, Ruby Pai, Jin Pan, Nir Paran, Benjamin Perry, Maneesh Pingle, Jakir Pinjari, Boaz Politi, Ailsa Powell, Vladimir Psenak, Reut Puni, Victor L. Rangel, Rambabu N. Reddi, St Patrick Reid, Efrat Resnick, Emily Grace Ripka, Matthew C. Robinson, Ralph P. Robinson, Jaime Rodriguez-Guerra, Romel Rosales, Dominic Rufa, Kadi Saar, Kumar Singh Saikatendu, Chris Schofield, Mikhail Shafeev, Aarif Shaikh, Jiye Shi, Khriesto Shurrush, Sukrit Singh, Assa Sittner, Rachael Skyner, Adam Smalley, Bart Smeets, Mihaela D. Smilova, Leonardo J. Solmesky, John Spencer, Claire Strain-Damerell, Vishwanath Swamy, Hadas Tamir, Rachael Tennant, Warren Thompson, Andrew Thompson, Susana Tomasio, Igor S. Tsurupa, Anthony Tumber, Ioannis Vakonakis, Ronald P. van Rij, Laura Vangeel, Finny S. Varghese, Mariana Vaschetto, Einat B. Vitner, Vincent Voelz, Andrea Volkamer, Frank von Delft, Annette von Delft, Martin Walsh, Walter Ward, Charlie Weatherall, Shay Weiss, Kris M. White, Conor Francis Wild, Matthew Wittmann, Nathan Wright, Yfat

Yahalom-Ronen, Daniel Zaidmann, Hadeer Zidane, and Nicole Zitzmann. Open science discovery of oral non-covalent sars-cov-2 main protease inhibitor therapeutics. *bioRxiv*, 2022.

[9] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13260–13271. Curran Associates, Inc., 2020.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

[11] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[12] Miguel García-Ortegón, Gregor NC Simm, Austin J Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. Dockstring: easy molecular docking yields better benchmarks for ligand design. *arXiv preprint arXiv:2110.15486*, 2021.

[13] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713. PMLR, 10–15 Jul 2018.

[14] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

[15] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

[16] Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V Chawla. Few-shot graph learning for molecular property prediction. *arXiv preprint arXiv:2102.07916*, 2021.

[17] Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M Brockway, and Alán Aspuru-Guzik. The harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*, 2(17):2241–2251, 2011.

[18] Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[19] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

[20] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Leveraging the feature distribution in transfer-based few-shot learning. In *International Conference on Artificial Neural Networks*, pages 487–499. Springer, 2021.

[21] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[22] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang Dong Yoo. Edge-labeling graph neural network for few-shot learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20, 2019.

[23] Minyoung Kim and Timothy Hospedales. Gaussian process meta few-shot classifier learning via linear discriminant laplace approximation. *arXiv preprint arXiv:2111.05392*, 2021.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Gregory R. Koch. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[26] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[27] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[28] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[29] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512, 2020.

[30] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sungju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations*, 2019.

[31] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1540–1552. PMLR, 26–28 Aug 2020.

[32] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2952–2960, New York, New York, USA, 20–22 Jun 2016. PMLR.

[33] Łukasz Maziarka, Tomasz Danel, Sławomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanisław Jastrzębski. Molecule attention transformer. *arXiv preprint arXiv:2002.08264*, 2020.

[34] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

[35] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.

[36] Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1996.

[37] Sebastian W. Ober, Carl E. Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1206–1216. PMLR, 27–30 Jul 2021.

[38] Eunbyung Park and Junier B Oliva. Meta-curvature. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[39] Massimiliano Patacchiola, John Bronskill, Aliaksandra Shysheya, Katja Hofmann, Sebastian Nowozin, and Richard E Turner. Contextual squeeze-and-excitation for efficient few-shot image classification. *arXiv preprint arXiv:2206.09843*, 2022.

[40] Massimiliano Patacchiola, Jack Turner, Elliot J. Crowley, Michael O' Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16108–16118. Curran Associates, Inc., 2020.

[41] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 737–746, New York, New York, USA, 20–22 Jun 2016. PMLR.

[42] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[43] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[44] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural networks*, 18(8):1093–1110, 2005.

[45] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, January 2006.

[46] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[47] Warren S. Sarle. Stopped training and other remedies for overfitting. In *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, pages 352–360, 1995.

[48] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[49] Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. *arXiv preprint arXiv:2007.10417*, 2020.

[50] Megan Stanley, John F Bronskill, Krzysztof Maziarz, Hubert Misztela, Jessica Lanini, Marwin Segler, Nadine Schneider, and Marc Brockschmidt. Fs-mol: A few-shot learning dataset of molecules. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[51] Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

[52] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702.e13, 2020.

[53] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[54] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020.

[55] Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning. *arXiv preprint arXiv:1905.12131*, 2019.

[56] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[57] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

[58] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[59] Yaqing Wang, Abulikemu Abuduweili, Quanming Yao, and Dejing Dou. Property-aware relation networks for few-shot molecular property prediction. *Advances in Neural Information Processing Systems*, 34:17441–17454, 2021.

[60] Oren F. Webb, Tommy J. Phelps, Paul R. Bienkowski, Philip M. Digrazia, David C. White, and Gary S. Sayler. Enzyme nomenclature, 1992.

[61] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[62] Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, 2016.

[63] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 09–11 May 2016. PMLR.

[64] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[65] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ArXiv*, abs/1810.00826, 2019.

[66] Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. idarts: Differentiable architecture search with stochastic implicit gradients. In *International Conference on Machine Learning*, pages 12557–12566. PMLR, 2021.

A Cauchy's Implicit Function Theorem

We state Cauchy's Implicit Function Theorem (IFT) in the context of ADKF-IFT in Theorem 1.

Theorem 1 (Implicit Function Theorem (IFT)) *Let $\mathcal{T}'$ be any given task. Suppose for some $\boldsymbol{\psi}'_{meta}$ and $\boldsymbol{\psi}'_{adapt}$ that $\frac{\partial \mathcal{L}_T(\boldsymbol{\psi}_{meta}, \boldsymbol{\psi}_{adapt}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{adapt}}\Big|_{\boldsymbol{\psi}'_{meta}, \boldsymbol{\psi}'_{adapt}} = \mathbf{0}$. Suppose that $\frac{\partial \mathcal{L}_T}{\partial \boldsymbol{\psi}_{adapt}}(\boldsymbol{\psi}_{meta}, \boldsymbol{\psi}_{adapt}, \mathcal{S}_{\mathcal{T}'}) : \Psi_{meta} \times \Psi_{adapt} \to \Psi_{adapt}$ is a continuously differentiable function w.r.t. $\boldsymbol{\psi}_{meta}$ and $\boldsymbol{\psi}_{adapt}$, and the Hessian $\frac{\partial^2 \mathcal{L}_T(\boldsymbol{\psi}_{meta}, \boldsymbol{\psi}_{adapt}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{adapt} \partial \boldsymbol{\psi}_{adapt}^T}\Big|_{\boldsymbol{\psi}'_{meta}, \boldsymbol{\psi}'_{adapt}}$ is invertible. Then, there exists an open set $U \in \Psi_{meta}$ containing $\boldsymbol{\psi}'_{meta}$ and a function $\boldsymbol{\psi}^*_{adapt}(\boldsymbol{\psi}_{meta}, \mathcal{S}_{\mathcal{T}'}) : \Psi_{meta} \to \Psi_{adapt}$, such that $\boldsymbol{\psi}'_{adapt} = \boldsymbol{\psi}^*_{adapt}(\boldsymbol{\psi}'_{meta}, \mathcal{S}_{\mathcal{T}'})$ and $\frac{\partial \mathcal{L}_T(\boldsymbol{\psi}_{meta}, \boldsymbol{\psi}_{adapt}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{adapt}}\Big|_{\boldsymbol{\psi}''_{meta}, \boldsymbol{\psi}^*_{adapt}(\boldsymbol{\psi}''_{meta}, \mathcal{S}_{\mathcal{T}'})} = \mathbf{0}$, $\forall\, \boldsymbol{\psi}''_{meta} \in U$. Moreover, the rate at which $\boldsymbol{\psi}^*_{adapt}(\boldsymbol{\psi}_{meta}, \mathcal{S}_{\mathcal{T}'})$ is changing w.r.t. $\boldsymbol{\psi}_{meta}$ for any $\boldsymbol{\psi}''_{meta} \in U$ is given by*

$$\begin{aligned}
&\frac{\partial \boldsymbol{\psi}^*_{adapt}(\boldsymbol{\psi}_{meta}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{meta}}\Big|_{\boldsymbol{\psi}''_{meta}} \\
&= -\left(\frac{\partial^2 \mathcal{L}_T(\boldsymbol{\psi}_{meta}, \boldsymbol{\psi}_{adapt}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{adapt} \partial \boldsymbol{\psi}_{adapt}^T}\right)^{-1} \frac{\partial^2 \mathcal{L}_T(\boldsymbol{\psi}_{meta}, \boldsymbol{\psi}_{adapt}, \mathcal{S}_{\mathcal{T}'})}{\partial \boldsymbol{\psi}_{adapt} \partial \boldsymbol{\psi}_{meta}^T}\Big|_{\boldsymbol{\psi}''_{meta}, \boldsymbol{\psi}^*_{adapt}(\boldsymbol{\psi}''_{meta}, \mathcal{S}_{\mathcal{T}'})}.
\end{aligned}$$

B Related Work

ADKF-IFT is part of a growing body of literature of techniques to train deep kernel GPs. As discussed in Section 3.3, ADKF-IFT *generalizes* DKL [63] and DKT [40], which exclusively use single-task learning and meta-learning, respectively. [29] and [57] propose adding intelligent regularization terms to the loss of DKL in order to mitigate overfitting. These works are better viewed as *complementary* to ADKF-IFT rather than *alternatives*: their respective regularization terms could easily be added to $\mathcal{L}_T$ in Equation (2) to improve performance. However, it is worth noting that the regularization strategies in both of these papers are designed for continuous inputs only, limiting their applicability to structured data like molecules. Finally, the preprint of [55] proposes an alternative adaptive deep kernel GP trained with meta-learning, where adaptation is performed by conditioning the feature extractor on an embedding of the entire support set instead. However, their empirical results were generally poor, and the method appeared very prone to overfitting; see Appendix J.

ADKF-IFT can also be viewed as a meta-learning algorithm comparable to many previously-proposed methods [26, 58, 13, 56, 38, 54, 20, 59, 5, 39]. One distinguishing feature of ADKF-IFT is that it is specially designed for deep kernel GPs, whereas most methods from computer vision are designed exclusively for neural network models, which as previously stated are unsuitable when reliable uncertainty estimates are required. Furthermore, many of these algorithms such as ProtoNet [48] are designed principally or exclusively for classification, while ADKF-IFT is suited to both regression and classification. Compared to model-agnostic frameworks like MAML [10], ADKF-IFT does not require coarse approximations of the hypergradient due to its use of the implicit function theorem. Finally, we note that the implicit function theorem employed in our work has been used in many previous machine learning papers in various contexts, e.g., neural architecture search [66], hyperparameter-tuning [2, 32, 41, 31, 7], and meta-learning [43, 27, 6].

C General Configurations of ADKF-IFT for Few-shot Learning Experiments

In this paper, we consider the specific instantiation of ADKF-IFT from Section 3.4. We solve the inner optimization problem (2) using the L-BFGS optimizer [28], since we choose $A_{\Psi_{\text{adapt}}} = A_\Theta$ and L-BFGS is the default choice for optimizing base kernel parameters in the literature. For the outer optimization problem (1), we approximate the expected hypergradient over $p(\mathcal{T})$ by averaging the hypergradients for a batch of K randomly sampled training tasks at each step, and update the meta-learned parameters $\boldsymbol{\psi}_{\text{meta}}$ with the averaged hypergradient using the Adam optimizer [24]. For all experiments on FS-Mol, we evaluate the performance of our model on a small set of validation tasks during meta-training and use early stopping [42] to avoid overfitting of $\boldsymbol{\psi}_{\text{meta}}$.

Table 3: Statistics of four few-shot molecular property prediction benchmarks from MoleculeNet.

Statistic	MoleculeNet benchmark task			
	Tox21	SIDER	MUV	ToxCast
#compounds	8,014	1,427	93,127	8,615
#tasks	12	27	17	617
#training tasks	9	21	12	450
#test tasks	3	6	5	167

We use zero mean function and set Matérn52 without automatic relevance determination (ARD) [36] as the base kernel in ADKF-IFT, since the typical sizes of the support sets in few-shot learning are too small to adjust a relatively large number of ARD lengthscales in ADKF-IFT. The lengthscale in the base kernel of ADKF-IFT is initialized using the median heuristic [14] for each task, with a log-normal prior centered at the initialization. Following [40], we treat binary classification as ± 1 label regression for ADKF-IFT.

D Details of MoleculeNet Benchmark Tasks

In Table 3, we summarize the four few-shot molecular property classification benchmark tasks (Tox21, SIDER, MUV, and ToxCast) from MoleculeNet [64] considered in Section 4.1.

E Detailed Configurations of All Compared Methods for FS-Mol

Single-task Methods. Single-task methods (RF, kNN, GP-ST, GNN-ST, and DKL) are trained separately on the support set of each test task, without leveraging the knowledge contained in the training tasks. The implementations of RF, kNN, and GNN-ST are taken from [50]. RF, kNN, and GP-ST operates on top of manually curated features obtained using RDKit. RF and kNN use extended connectivity fingerprint [46] (count-based fingerprint with radius 2 and size 2,048) and phys-chem descriptors (with size 42). GP-ST uses fingerprint (with radius 2 and 2,048 bits based on count simulation). DKL operates on top of a combination of extended connectivity fingerprint [46] (count-based fingerprint with radius 2 and size 2,048) and features extracted by a GNN. We use the same base kernel in DKL as in ADKF-IFT. DKL is trained for 50 epochs on the support set of each test task. Hyperparameter search configurations for these methods are based on the extensive industrial experience from the authors of [50]. GNN-ST uses a GNN with a hidden dimension of 128 and a gated readout function [15], considering ~ 30 hyperparameter search configurations.

Multi-task Pretraining. The implementation of GNN-MT is taken from [50]. GNN-MT shares a GNN with a hidden dimension of 128 using principal neighborhood message aggregation [9] across tasks, and uses a task-specific gated readout function [15] and an MLP with one hidden layer on top for each individual task. The model is trained on the support sets of all training tasks with early stopping based on the validation performance on the validation tasks. The task-specific components of the model are fine-tuned for each test task.

Self-supervised Pretraining. The implementation of MAT is taken from [50]. We use the official pretrained model parameters [33], which is pretrained on 2 millions molecules sampled from the ZINC15 dataset [51]. We fine-tuned it for each test task with hyperparameter search and early stopping based on 20% of the support set for each task.

Meta-learning Methods. Meta-learning methods (PAR, ProtoNet, GNN-MAML, CNP, DKT, and ADKF-IFT) enable knowledge transfer among related small datasets. The implementations of ProtoNet and GNN-MAML are taken from [50]. The implementation of PAR is taken from its official implementation and integrated into the FS-Mol training and evaluation pipeline. PAR, ProtoNet, CNP, DKT, and ADKF-IFT operate on top of a combination of extended connectivity fingerprint [46] (count-based fingerprint with radius 2 and size 2,048) and features extracted by a GNN. The feature extractor architecture used for DKL, PAR, CNP, DKT, and ADKF-IFT is the same as that used for ProtoNet, GNN-MAML, GNN-ST, and GNN-MT in [50], with the size of the feature representation being tuned on the validation tasks. We use the same base kernel in DKT as in ADKF-IFT.

Table 4: Mean ranks of all compared methods in terms of their performance on all FS-Mol test tasks.

(a) Classification (157 tasks).

Method	Support set size				
	16	32	64	128	256
GNN-ST	11.29	11.53	11.75	11.85	12.19
kNN	10.89	10.48	10.33	10.15	9.37
MAT	10.43	10.44	10.19	9.69	9.70
RF	8.15	7.89	7.06	6.25	4.47
PAR	7.70	7.98	8.30	8.83	10.81
GNN-MT	7.33	7.18	7.08	6.59	6.53
DKL	7.28	7.49	7.98	8.42	8.21
GP-ST	6.71	6.57	6.28	6.18	5.14
GNN-MAML	6.36	6.92	7.42	7.89	8.90
CNP	5.00	5.81	6.36	6.91	7.78
ProtoNet	4.00	3.40	3.11	2.98	3.85
DKT	3.44	3.19	2.99	2.99	2.67
ADKF-IFT	**2.41**	**2.12**	**2.14**	**2.26**	**1.38**

(b) Regression (111 tasks).

Method	Support set size				
	16	32	64	128	256
MAT	7.60	7.45	7.26	7.06	7.19
GNN-MT	6.61	6.40	6.15	5.95	5.58
RF	5.00	4.47	4.16	3.72	3.56
DKL	4.42	5.16	5.63	6.10	6.35
GP-ST	4.23	4.14	3.87	3.37	3.07
CNP	3.88	4.45	4.95	5.73	6.47
DKT	**2.12**	2.08	2.29	2.32	2.43
ADKF-IFT	**2.12**	**1.86**	**1.68**	**1.74**	**1.36**

F Task-level Evaluation Metrics for FS-Mol

Binary Classification. Following [50], the task-level metric used for the binary classification task in FS-Mol is change in area under the precision-recall curve (ΔAUPRC), which is sensitive to the balance of the two classes in the query sets and allows for a comparison to the performance of a random classifier:

$$\begin{aligned}\Delta\text{AUPRC(target classifier, } \mathcal{T}) &= \text{AUPRC(target classifier, } \mathcal{Q}_\mathcal{T}) - \text{AUPRC(random classifier, } \mathcal{Q}_\mathcal{T}) \\ &= \text{AUPRC(target classifier, } \mathcal{Q}_\mathcal{T}) - \frac{\text{\#positive data points in } \mathcal{Q}_\mathcal{T}}{N_{\mathcal{Q}_\mathcal{T}}}.\end{aligned}$$

Regression. We propose to use the predictive/out-of-sample coefficient of determination (R^2_{os}) as the task-level metric for the regression task in FS-Mol, which takes into account forecast errors:

$$R^2_{os}(\text{target regressor } g, \mathcal{T}) = 1 - \frac{\sum_{(\mathbf{x}_m, y_m)\in\mathcal{Q}_\mathcal{T}} (y_m - g(\mathbf{x}_m))^2}{\sum_{y_m \in \mathcal{Q}^y_\mathcal{T}} (y_m - \bar{y}_{\mathcal{S}_\mathcal{T}})^2},$$

where $\bar{y}_{\mathcal{S}_\mathcal{T}} = \frac{1}{N_{\mathcal{S}_\mathcal{T}}} \sum_{y_n \in \mathcal{S}^y_\mathcal{T}} y_n$ is the mean target value in the support set $\mathcal{S}_\mathcal{T}$. This is different from the regular coefficient of determination (R^2), wherein the total sum of squares in the denominator are computed using the mean target value $\bar{y}_{\mathcal{Q}_\mathcal{T}} = \frac{1}{N_{\mathcal{Q}_\mathcal{T}}} \sum_{y_m \in \mathcal{Q}^y_\mathcal{T}} y_m$ in the query set $\mathcal{Q}_\mathcal{T}$.

G Further Experimental Results on FS-Mol

G.1 Overall Performance

Table 4 shows that ADKF-IFT achieves the best mean rank for both classification and regression tasks at all considered support set sizes. The trends of these mean ranks are consistent to those in Figure 2. Figures 6 and 7 show the box plots for the classification and regression performances of all compared methods on all FS-Mol test tasks, respectively. These plots are a disaggregated representation of the results in Figure 2.

G.2 Statistical Comparison

Table 5 shows the p-values from the two-sided Wilcoxon signed-rank test for statistical comparisons between ADKF-IFT and the next second method, namely DKT. The test results indicate that their median performance difference is nonzero (i.e., ADKF-IFT significantly outperforms DKT) for the classification task at all considered support set sizes and for the regression task at support set sizes 64, 128, and 256 (at significance level $\alpha = 0.05$).

G.3 Sub-benchmark Performance

The tasks in FS-Mol can be partitioned into 7 sub-benchmarks by Enzyme Commission (EC) number [60], which enables sub-benchmark evaluation within the entire benchmark. Ideally, the best method

Table 5: p-values from the two-sided Wilcoxon signed-rank test for statistical comparisons between ADKF-IFT and DKT. The null hypothesis is that the median of their performance differences on all FS-Mol test tasks is zero. The significance level is set to $\alpha = 0.05$.

FS-Mol learning task	Support set size				
	16	32	64	128	256
Classification (157 tasks)	$\mathbf{1.4 \times 10^{-12}}$	$\mathbf{8.1 \times 10^{-14}}$	$\mathbf{2.3 \times 10^{-12}}$	$\mathbf{1.0 \times 10^{-8}}$	$\mathbf{3.4 \times 10^{-7}}$
Regression (111 tasks)	8.2×10^{-2}	9.6×10^{-2}	$\mathbf{3.7 \times 10^{-5}}$	$\mathbf{7.1 \times 10^{-5}}$	$\mathbf{9.8 \times 10^{-7}}$

Table 6: Mean performance with standard errors of top performing methods on FS-Mol test tasks within each sub-benchmark (broken down by EC category) at support set size 64 (the median of all considered support sizes). Note that class 2 is most common in the FS-Mol training set ($\sim 1,500$ training tasks), whereas classes 6 and 7 are least common in the FS-Mol training set (< 50 training tasks each).

(a) Classification (ΔAUPRC).

FS-Mol sub-benchmark (EC category)			Method				
Class	Description	#tasks	RF	GP-ST	ProtoNet	DKT	ADKF-IFT
1	oxidoreductases	7	0.156 ± 0.044	0.152 ± 0.040	0.137 ± 0.037	0.145 ± 0.040	$\mathbf{0.160 \pm 0.045}$
2	kinases	125	0.152 ± 0.009	0.161 ± 0.009	0.285 ± 0.010	0.282 ± 0.010	$\mathbf{0.299 \pm 0.010}$
3	hydrolases	20	0.229 ± 0.032	0.230 ± 0.032	0.245 ± 0.034	0.254 ± 0.034	$\mathbf{0.262 \pm 0.033}$
4	lysases	2	0.276 ± 0.182	$\mathbf{0.284 \pm 0.189}$	0.265 ± 0.211	0.272 ± 0.206	0.279 ± 0.201
5	isomerases	1	0.166 ± 0.040	$\mathbf{0.212 \pm 0.052}$	0.172 ± 0.044	0.204 ± 0.058	0.198 ± 0.046
6	ligases	1	0.149 ± 0.035	0.199 ± 0.028	0.170 ± 0.028	0.229 ± 0.013	$\mathbf{0.231 \pm 0.022}$
7	translocases	1	$\mathbf{0.128 \pm 0.039}$	0.109 ± 0.049	0.099 ± 0.028	0.122 ± 0.022	0.109 ± 0.033
	all enzymes	157	0.163 ± 0.009	0.171 ± 0.009	0.271 ± 0.009	0.271 ± 0.010	$\mathbf{0.285 \pm 0.010}$

(b) Regression (R^2_{os}).

FS-Mol sub-benchmark (EC category)			Method				
Class	Description	#tasks	RF	GP-ST	CNP	DKT	ADKF-IFT
1	oxidoreductases	6	0.108 ± 0.087	0.103 ± 0.076	-0.012 ± 0.011	0.098 ± 0.078	$\mathbf{0.116 \pm 0.079}$
2	kinases	82	0.160 ± 0.019	0.162 ± 0.022	0.127 ± 0.017	0.343 ± 0.022	$\mathbf{0.363 \pm 0.024}$
3	hydrolases	19	0.256 ± 0.058	0.267 ± 0.061	0.014 ± 0.015	0.295 ± 0.063	$\mathbf{0.310 \pm 0.062}$
4	lysases	2	0.418 ± 0.405	0.417 ± 0.416	0.100 ± 0.068	0.440 ± 0.418	$\mathbf{0.442 \pm 0.403}$
5	isomerases	1	0.125 ± 0.077	0.086 ± 0.082	-0.012 ± 0.010	0.209 ± 0.113	$\mathbf{0.226 \pm 0.063}$
6	ligases	1	0.182 ± 0.040	0.202 ± 0.079	0.002 ± 0.004	0.277 ± 0.035	$\mathbf{0.279 \pm 0.043}$
	all enzymes	111	0.178 ± 0.019	0.181 ± 0.021	0.097 ± 0.014	0.321 ± 0.021	$\mathbf{0.340 \pm 0.022}$

should be able to perform well across all sub-benchmarks. Table 6 shows the test performance of top performing methods on all sub-benchmarks at support set size 64 (the median of all considered support sizes) for both the classification and regression tasks. The results indicate that, in addition to achieving best overall performance, ADKF-IFT achieves the best performance on all sub-benchmarks for the regression task and on more than half of the sub-benchmarks for the classification task.

G.4 Meta-testing Costs

Figure 5 shows the meta-testing costs of all compared meta-learning methods in terms of wall-clock time[1] on a pre-defined set of FS-Mol classification tasks. These experiments are run on a single NVIDIA GeForce RTX 2080 Ti. It can be seen that ADKF-IFT is ~ 2.5x slower than CNP, ProtoNet, and DKT, but still much faster than GNN-MAML. We did not report the wall-clock time for PAR, because it is extremely memory intensive (PAR takes > 10x memory than ADKF-IFT does) and thus cannot be run on a GPU. We stress that this is not an important metric for this paper, as real-time adaptation is not required in drug discovery applications, but could be of interest if ADKF-IFT were to be deployed in other settings.

[1]We acknowledge that wall-clock time may not be the best metric for measuring the costs, since some meta-learning methods could be parallelized, which will reduce the wall-clock time accordingly. An alternative metric is multiply–accumulate operation (MAC). However, it is difficult to obtain the accurate number of MACs due to the opaqueness of the GP modules used.

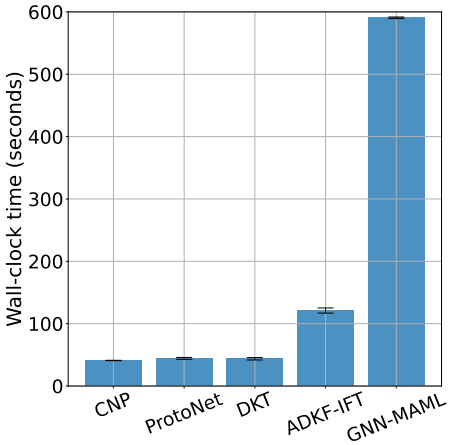


Figure 5: Wall-clock time consumed (with standard errors) when meta-testing on a pre-defined set of FS-Mol classification tasks using each of the compared meta-learning methods.

Table 7: Descriptions of four out-of-domain molecular design tasks.

Molecular design task	Data source	#compounds	Target	Target source
Molecular docking (ESR2)	DOCKSTRING training set	2,312	binding score	AutoDock Vina
Antibiotic discovery (E. coli BW25113)	Antibiotic training set	2,335	relative growth	screening
Antiviral drug design (SARS-CoV-2)	COVID Moonshot	1,926	pIC50 Fluorescence	experimental lab
Material design (Organic Photovoltaic)	Harvard Clean Energy Project	2,012	power conversion efficiency	DFT simulation

H Details of the Out-of-domain Molecular Optimization Experiments

In Table 7, we summarize the four molecular design tasks considered in Section 4.3. Note that the datasets for the molecular docking and material design tasks are subsampled from the much larger datasets provided in DOCKSTRING [12] and Harvard Clean Energy Project [17], respectively. The datasets for the antibiotic discovery and antiviral drug design tasks are taken from the antibiotic training set and the COVID Moonshot dataset provided in [52] and [8], respectively.

For the configuration of the GP, we use the Tanimoto kernel for fingerprint (with radius 2 and 2,048 bits based on count simulation) and Matérn52 kernel without ARD (with a log-normal prior over the lengthscale, centered at the median heuristic initialization) for all the other compared feature representations. We re-fit the base kernel parameters using all available data points at the beginning of each BO iteration.

I Discussions of Methods for Learning Deep Kernel GPs

I.1 The Overfitting Issue in DKL

Deep Kernel Learning (DKL) [63] is a single-task method for fitting a deep kernel to a dataset. DKL jointly fits both the feature extractor parameters ϕ and base kernel parameters $\boldsymbol{\theta}$ by maximizing the GP marginal likelihood on a single dataset (i.e., DKL essentially fits a neural network with a GP "head" to a dataset).

It is well known that neural networks will easily overfit to small datasets [47]. This overfitting also happens in DKL, despite the fact that it fits the neural network parameters using a type-II maximum likelihood approach: although early DKL papers suggested that the "model complexity" term (as measured by the log determinant of the kernel matrix) in the GP marginal likelihood objective would prevent this overfitting from happening [63], recent follow-up work showed that this is not the case [37] – a deep-kernel GP can simultaneously overfit to the training data and appear to have a low "model complexity".

I.2 The Underfitting Issue in DKT

Deep Kernel Transfer (DKT) [40] is a meta-learning method for fitting a deep kernel to a distribution of datasets. DKT jointly fits both the feature extractor parameters $\boldsymbol{\phi}$ and base kernel parameters $\boldsymbol{\theta}$ by maximizing the expected GP marginal likelihood over a distribution of datasets.

To mitigate the overfitting issue of DKL using meta-learning, DKT makes a very strong assumption that different tasks in the task distribution are drawn from an identical GP prior over functions. Explicitly, this means that the data generating process is assumed to have the same noise level, same amplitude, and same "characteristic lengthscale" for every task in the meta-dataset, which is a very restrictive assumption violated by most real-world problems. For example, different datasets in a meta-dataset may have

- highly varying noise levels, so modelling all tasks with the same amount of observation noise will not be realistic;
- different output ranges and units for regression: for example, one task might have data in the range 1-20 μM, while another might have 0-100$\%$ inhibition, meaning that a single signal variance (kernel amplitude) will not model the data well;
- different "characteristic lengthscales": for some tasks, structurally similar molecules have very strongly correlated output labels, while for other tasks it is much weaker (i.e., there is much more variation in the labels of very similar molecules), suggesting that the "characteristic lengthscale" will be different.

Inevitably, trying to fit such a misspecified model will result in a set of compromised base kernel parameters $\boldsymbol{\theta}$ which fit all datasets okay on average but do not fit each individual dataset very well. This is the underfitting issue of DKT.

I.3 The Advantage of ADKF-IFT

ADKF-IFT combines DKL and DKT in a way that can potentially inherit the strengths of both methods and the weaknesses of neither. By adapting the base kernel parameters $\boldsymbol{\theta}$ specifically to each task, it prevents underfitting due to varying ranges, lengthscales, or noise levels between datasets. By meta-learning the feature extractor on many datasets, it prevents overfitting as observed by [37]. This advantage is both *theoretically principled* (by solving a bilevel optimization objective using the implicit function theorem) and *empirically observable* (we showed a statistically significant performance improvement of ADKF-IFT over DKT in Section 4.2).

J Comments on ADKL-GP

The training objective for ADKL-GP [55] is equivalent to the objective for DKT with an added contrastive loss, weighted by a sensitive hyperparameter γ (see Equation (13) in their preprint [55]). γ can be interpreted as balancing the degree of regularization between two extremes:

1. If $\gamma = 0$, there is no regularization of the task encoding network, making significant overfitting to the meta-dataset possible. This is effectively equivalent to standard DKL [63].
2. As $\gamma \to \infty$, the regularization becomes infinitely strong, causing the task embeddings $\mathbf{z}_\mathcal{T}$ to collapse, and thereby preventing them from transmitting any information about specific datasets. With no information from $\mathbf{z}_\mathcal{T}$ in this case, the objective is essentially the same as that of DKT [40].

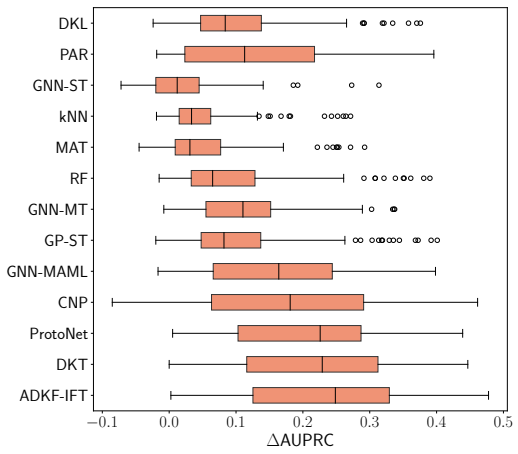
For this method to be an informative baseline, γ would need to be appropriately tuned so that the method can be distinguished from DKL and DKT. Unfortunately, their preprint [55] contains little guidance on how to perform this tuning. They perform a grid search over all hyperparameters including $\gamma \in \{0, 0.01, 0.1\}$ but find no consistent trend besides $\gamma > 0$ being slightly helpful, although the differences in performance were small. Due to computational constraints, we were unable to run extensive experiments tuning γ and the set encoder architecture. Our preliminary results show that the performance of ADKL-GP is much worse than that of our ADKF-IFT, but we feel that it would be unfair to report such preliminary results that were not tuned and checked as thoroughly as the other baselines. Therefore, we chose not to report the results of ADKL-GP in our experiments.

Even if a carefully-tuned implementation of ADKL-GP was able to outperform our implementation of ADKF-IFT, we do not believe that would diminish the value of our contribution. We see the lack of tunable regularization hyperparameters in ADKF-IFT as a significant *strength*, because it makes our method more consistent and reliable. Furthermore, our ADKF-IFT is able to adapt important parameters such as likelihood noise level, kernel lengthscales, and signal variance (kernel amplitude), which will almost certainly vary among datasets but cannot be adapted using the ADKL-GP method. For all these reasons, we hope that the reader understands and agrees with our decision to omit results for ADKL-GP from our experiments in Section 4.

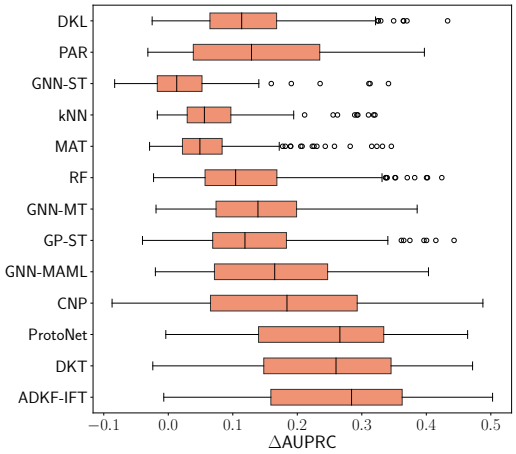
K Conclusion and Future Work

We have proposed Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT), a novel framework for fitting deep kernels that interpolates between meta-learning and conventional deep kernel learning. ADKF-IFT meta-learns a feature extractor across tasks such that the task-specific GP models estimated on top of the extracted feature representations can achieve lowest possible prediction error on average. ADKF-IFT is implemented by solving a bilevel optimization objective via implicit differentiation. We have shown that ADKF-IFT is a unifying framework containing DKL and DKT as special cases. We have demonstrated that ADKF-IFT learns generally useful feature representations, achieving state-of-the-art performance on a variety of real-world few-shot molecular property prediction tasks and on out-of-domain molecular property prediction and optimization tasks. We believe that ADKF-IFT could potentially be an important method to produce well-calibrated models for fully-automated high-throughput experimentation in the future.

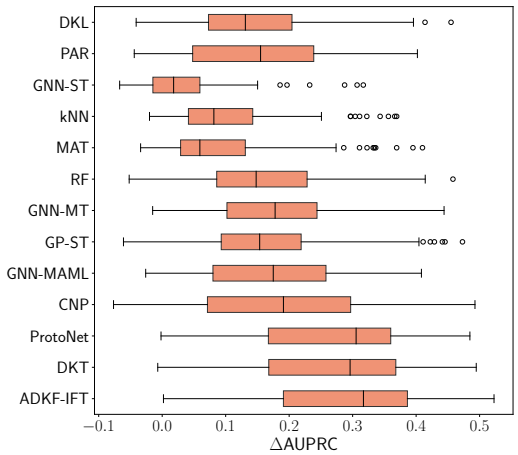
Some directions for future work are as follows: 1) using ARD in the base kernel so that feature selection for each individual task can be done by the GP model, with potential overfitting problems being reduced by assuming a sparse prior over lengthscales or by learning a low-dimensional manifold for them; 2) adapting the feature extractor to each task as well by allowing small deviations across tasks according to a meta-learned prior on the feature extractor parameters (e.g., as described in [6]); 3) adopting a more principled approximate inference strategy for few-shot GP classification (e.g., Pólya-Gamma data augmentation [49] or Laplace approximation [23]); and 4) injecting domain expertise in drug discovery into the base kernel with hand-curated features and kernel combinations.



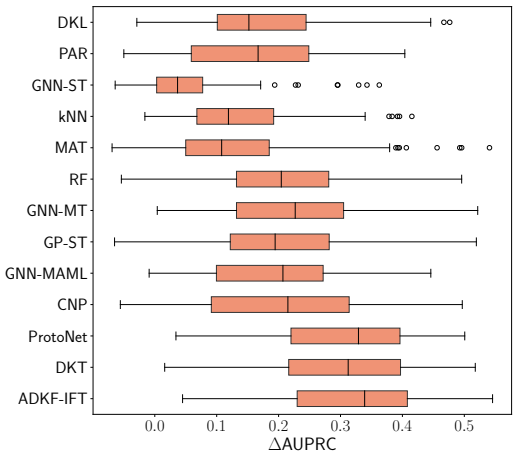
(a) $N_{\mathcal{S}_{\mathcal{T}_*}} = 16$.



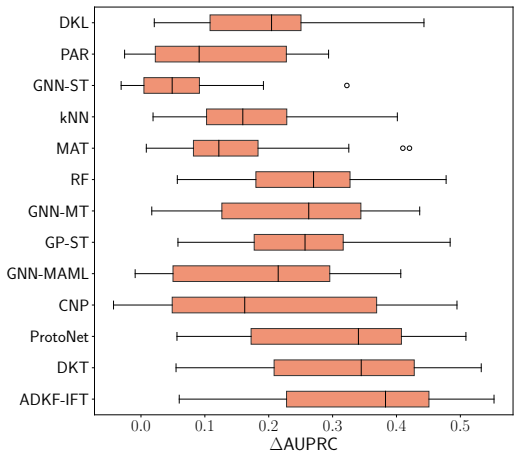
(b) $N_{\mathcal{S}_{\mathcal{T}_*}} = 32$.



(c) $N_{\mathcal{S}_{\mathcal{T}_*}} = 64$.

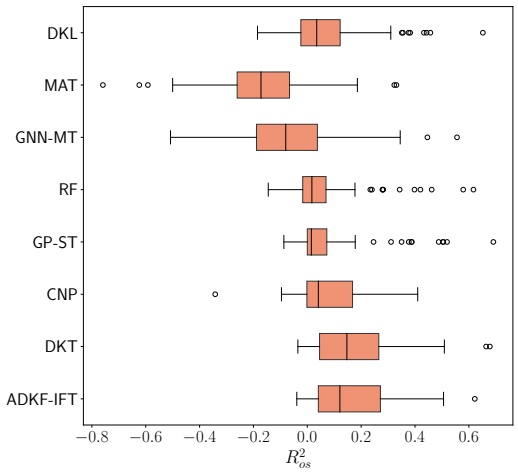


(d) $N_{\mathcal{S}_{\mathcal{T}_*}} = 128$.

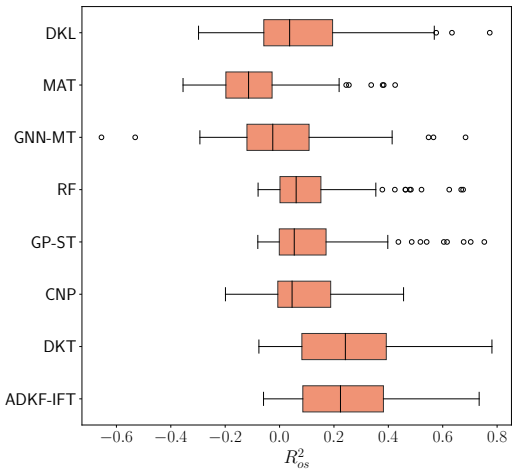


(e) $N_{\mathcal{S}_{\mathcal{T}_*}} = 256$.

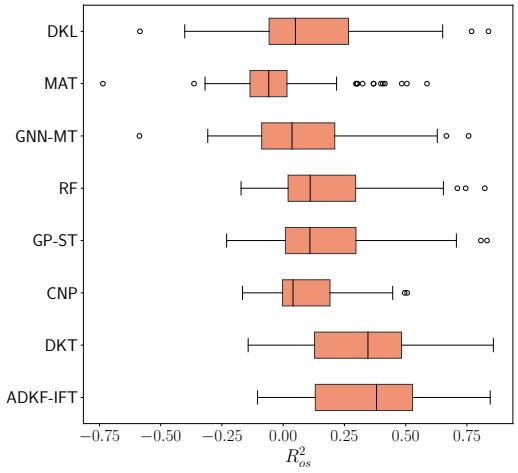
Figure 6: Box plots for the classification performance of all compared methods on 157 FS-Mol test tasks at different support set sizes.



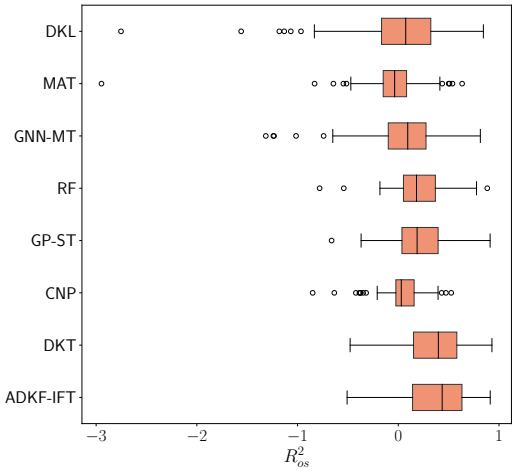
(a) $N_{\mathcal{S}_{\mathcal{T}_*}} = 16$.



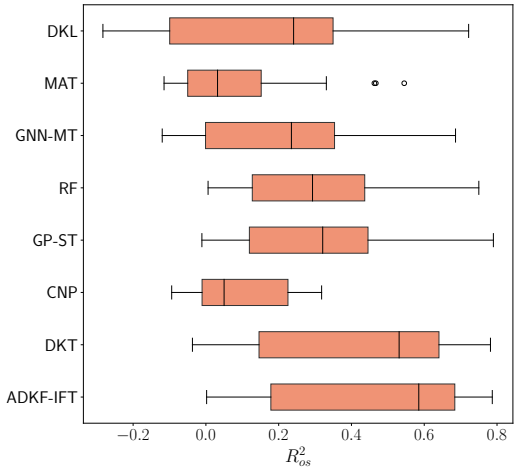
(b) $N_{\mathcal{S}_{\mathcal{T}_*}} = 32$.



(c) $N_{\mathcal{S}_{\mathcal{T}_*}} = 64$.



(d) $N_{\mathcal{S}_{\mathcal{T}_*}} = 128$.



(e) $N_{\mathcal{S}_{\mathcal{T}_*}} = 256$.

Figure 7: Box plots for the regression performance of all compared methods on 111 FS-Mol test tasks at different support set sizes.