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 kernels 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 Gaussian process 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 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 SOTA 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

A wide range of real-world applications require machine learning algorithms to make robust predictions with well-calibrated uncertainty given a small training set. One example is drug discovery, where the goal is to efficiently find novel molecules with specific physicochemical properties. Bayesian optimization (BO) is potentially useful for accelerating and automating this process, wherein data collection is guided by an acquisition function that uses both the predictions and uncertainty estimates provided by a surrogate model to balance exploration and exploitation [11]. The choice of the surrogate model in BO is thus vital. Although neural networks have been successful in making accurate predictions for molecular properties [15], they cannot effectively be used as surrogate models in BO, as they are notoriously non-robust and overconfident [44], especially when trained on small datasets: with few available measurements at the early stage of a drug discovery project, it is extremely hard to fit a neural network with good predictive performance, let alone good predictive uncertainty.

Meta-learning is often employed to improve the predictive performance of neural networks in the low-data regime by leveraging a large number of small datasets, which enables knowledge transfer among related low-data tasks. Surprisingly, in a recently proposed few-shot molecular property prediction benchmark FS-Mol [41], many popular meta-learning and self-supervised pretraining methods cannot even beat Random Forest (a single-task method trained separately on each task). The best performing baseline method in FS-Mol, namely ProtoNet [39], employs a neural network to meta-learn feature representations which are then fed into a nearest centroid classifier. Although ProtoNet ranks first among all FS-Mol baseline methods, its nearest centroid classifier is unsatisfactory in many aspects: 1) it lacks flexibility and tends to underfit when a large number of labeled examples are available (see Figure 1 for empirical evidence); 2) it does not have a clear extension to solving regression problems; and 3) it provides no predictive uncertainty. One way to address these limitations is to replace the nearest centroid classifier with a Gaussian process (GP) model [37]. This allows us to exploit the complementary strengths of neural networks and GPs for meta-learning, i.e., to combine neural networks' representation learning ability and GPs' robustness and uncertainty-awareness.

A popular way to combine neural networks and GPs is *deep kernel learning* [17, 55], wherein a kernel is constructed by first embedding data into a low-dimensional feature representation space using a neural network, then applying a "base kernel" (e.g., a Matérn kernel) to these feature representations. However, there is not a consensus on how to train these models on small datasets. Although early results suggested that the parameters of both the neural network and base kernel could be tuned by maximizing the GP marginal likelihood (the *conventional* way of training deep kernel GPs in the single-task setting [55]), subsequent work has shown that this can often lead to overfitting and a collapse in the GP's uncertainty estimates (i.e., losing GP's main benefits) [30]. This problem is especially acute with small datasets. Other researchers have proposed that the parameters in both parts can be jointly trained in a meta-learning framework to avoid overfitting [32]. These works do not report overfitting, but they assume that all datasets can be well-approximated using exactly the same hyperparameters for the base kernel. This assumption is unlikely to hold in practice: for example, different tasks may have different amounts of observation noise and different "characteristic lengthscales". Ultimately, neither of these strategies is a totally satisfactory way of training deep kernel GPs to transfer knowledge among small datasets.

In this work, we propose *Adaptive Deep Kernel Fitting with Implicit Function Theorem* (ADKF-IFT) to address the aforementioned limitations when fitting deep kernel GPs to small datasets:

- ADKF-IFT *generalizes* the conventional learning approach of [55] and the meta-learning approach of [32]: it partitions the parameters of a deep kernel into two disjoint sets, then training one set with meta-learning which facilitates conventional learning for the other set.
- ADKF-IFT meta-learns *generally useful feature representations*, such that task-specific GP models estimated on top achieve the lowest possible predictive error on average. The resulting nested optimization problem is solved using the implicit function theorem (IFT).
- ADKF-IFT is a *unified framework* containing DKL and DKT as special cases.
- While ADKF-IFT is a completely general method, we expect it to be especially well-suited for drug discovery problems (c.f. Section 4.1). We show that ADKF-IFT significantly outperforms previous SOTA methods on a variety of real-world few-shot molecular property prediction tasks and out-of-domain molecular property prediction and optimization tasks.

2 Background and Notation

2.1 Deep Kernel Gaussian Processes

Gaussian processes (GPs) [37] are tools for specifying priors over functions. 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 $p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})$ 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 $(\mathbf{X}_*, \mathbf{y}_*)$ (see [37] for details).

Deep kernel learning [17, 55] is a popular way to combine neural networks and GPs. A deep kernel is constructed by feeding the feature representations $\mathbf{h} = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x})$ and $\mathbf{h}' = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}')$ into a base kernel $c_{\boldsymbol{\theta}}(\mathbf{h}, \mathbf{h}')$, where $\mathbf{f}_{\boldsymbol{\phi}} : \mathcal{X} \to \mathbb{R}^n$ is a neural network feature extractor with learnable parameters $\boldsymbol{\phi} \in \Phi$. The resulting deep kernel function $k_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, \mathbf{x}') = c_{\boldsymbol{\theta}}(\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}'))$ can be used as covariance functions to specify deep kernel GP priors $p(f) = \mathcal{GP}(f; 0, k_{\boldsymbol{\theta},\boldsymbol{\phi}}(\cdot,\cdot))$.

2.2 Few-shot Learning

Few-shot learning [49] requires learning algorithms to be able to adapt to unseen tasks given few labeled examples. It often benefits from knowledge transfer among related low-data tasks, as single-task algorithms can easily overfit or underfit. Specifically, we are given a set of training tasks $\mathcal{D} = \{\mathcal{T}_t\}_{t=1}^{T}$ and some unseen test tasks $\mathcal{D}_* = \{\mathcal{T}_*\}$. Each task $\mathcal{T} = \{\mathcal{S}_{\mathcal{T}}, \mathcal{Q}_{\mathcal{T}}\}$ consists of a support set $\mathcal{S}_{\mathcal{T}} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_{\mathcal{S}_{\mathcal{T}}}}$ and a query set $\mathcal{Q}_{\mathcal{T}} = \{(\mathbf{x}_m, y_m)\}_{m=1}^{N_{\mathcal{Q}_{\mathcal{T}}}}$. The input data $\mathbf{x}$ belongs to the domain $\mathcal{X}$ (e.g., molecule space), and the target variable y is either a continuous variable (e.g., inhibitory concentration) or a discrete variable (e.g., active vs inactive). We denote the collections of all input data points and all targets in the support set $\mathcal{S}_\mathcal{T}$ by $\mathcal{S}_\mathcal{T}^{\mathbf{x}}$ and $\mathcal{S}_\mathcal{T}^{y}$, respectively. $\mathcal{Q}_\mathcal{T}^{\mathbf{x}}$ and $\mathcal{Q}_\mathcal{T}^{y}$ are similarly defined for the query set $\mathcal{Q}_\mathcal{T}$. Typically, a large number $T = |\,\mathcal{D}\,|$ of training tasks are available, each of which has a small number $N_{\mathcal{S}_\mathcal{T}}$ of labeled examples in its support set $\mathcal{S}_\mathcal{T}$. The goal of few-shot learning is to produce a model using the training tasks $\mathcal{D}$ such that it can be quickly adapted to any unseen test task $\mathcal{T}_*$, i.e., achieve low prediction error on the query set $\mathcal{Q}_{\mathcal{T}_*}$ after processing a small support set $\mathcal{S}_{\mathcal{T}_*}$ associated with $\mathcal{T}_*$.

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

This section describes the details of our proposed method, Adaptive Deep Kernel Fitting (ADKF-IFT). At its core, ADKF-IFT is a method for fitting a GP with a deep kernel function $k_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, \mathbf{x}')$ using a combination of meta-learning and conventional deep kernel learning. Let A_Θ and A_Φ be the sets of GP hyperparameters and feature extractor parameters, respectively. Denote the set of all parameters in a deep kernel by $A_\Psi = A_\Theta \cup A_\Phi$. ADKF-IFT requires that A_Ψ be partitioned into two disjoint sets $A_{\Psi_\text{meta}} \cup A_{\Psi_\text{adapt}} = A_\Psi$, where the parameters $\boldsymbol{\psi}_\text{meta}$ in A_{Ψ_meta} will be fit to a collection of tasks using meta-learning, and the parameters $\boldsymbol{\psi}_\text{adapt}$ in A_{Ψ_adapt} will be adapted separately for each task.

3.1 General Framework

ADKF-IFT is most easily understood by separately considering $\boldsymbol{\psi}_\text{meta}$ and $\boldsymbol{\psi}_\text{adapt}$. For a given task $\mathcal{T}$ and current meta-learned parameters $\boldsymbol{\psi}_\text{meta}$, the task-specific parameters $\boldsymbol{\psi}_\text{adapt}$ are chosen to minimize the *training loss* $\mathcal{L}_T$ evaluated on the task's support set $\mathcal{S}_\mathcal{T}$, see (2) below. That is, $\boldsymbol{\psi}_\text{adapt}$ is *adapted* to the support set $\mathcal{S}_\mathcal{T}$ of the task $\mathcal{T}$ under consideration during both meta-training and meta-testing. The resulting optimal value for $\boldsymbol{\psi}_\text{adapt}$ given $\boldsymbol{\psi}_\text{meta}$ and $\mathcal{S}_\mathcal{T}$ is called the *best response function*, denoted by $\boldsymbol{\psi}^*_\text{adapt}(\boldsymbol{\psi}_\text{meta}, \mathcal{S}_\mathcal{T})$. Knowing that $\boldsymbol{\psi}_\text{adapt}$ will be adapted separately to each task, the remaining problem is to choose a value for $\boldsymbol{\psi}_\text{meta}$ during meta-training. We propose to do this by minimizing the average *predictive validation loss* $\mathcal{L}_V$ evaluated on the query set $\mathcal{Q}_\mathcal{T}$ of a random training task $\mathcal{T} \sim p(\mathcal{T})$ given the optimal task-specific parameters $\boldsymbol{\psi}^*_\text{adapt}(\boldsymbol{\psi}_\text{meta}, \mathcal{S}_\mathcal{T})$, see (1) below, where $p(\mathcal{T})$ denotes the uniform distribution over training tasks. That is, $\boldsymbol{\psi}_\text{meta}$ is chosen such that GP with $\boldsymbol{\psi}_\text{adapt}$ adapted achieves the lowest possible average predictive 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 support set $\mathcal{S}_\mathcal{T}$. These objectives can be viewed jointly as forming the following *bilevel optimization* problem:

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

$$s.t. \quad \boldsymbol{\psi}^*_\text{adapt}(\boldsymbol{\psi}_\text{meta}, \mathcal{S}_\mathcal{T}) = \operatorname*{arg\,min}_{\boldsymbol{\psi}_\text{adapt}} \; \mathcal{L}_T(\boldsymbol{\psi}_\text{meta}, \boldsymbol{\psi}_\text{adapt}, \mathcal{S}_\mathcal{T}). \tag{2}$$

This formulates a meta-learning objective where we optimize the expected predictive performance on the query sets (in Equation (1)) after processing the corresponding support sets (in Equation (2)).

We propose to solve this bilevel optimization problem using gradient-based optimization techniques. For the inner optimization problem (2), the gradient of $\mathcal{L}_T$ w.r.t. $\boldsymbol{\psi}_\text{adapt}$ can be easily obtained by auto-differentiation. Therefore, for a given task $\mathcal{T}$ and current meta-learned parameters $\boldsymbol{\psi}_\text{meta}$, we can obtain $\boldsymbol{\psi}^*_\text{adapt}(\boldsymbol{\psi}_\text{meta}, \mathcal{S}_\mathcal{T})$ using, e.g., the L-BFGS optimizer [24]. However, for the outer optimization problem (1), it is less straightforward how to compute the gradient of $\mathcal{L}_V$ w.r.t. $\boldsymbol{\psi}_\text{meta}$. This is because $\boldsymbol{\psi}^*_\text{adapt}(\boldsymbol{\psi}_\text{meta}, \mathcal{S}_\mathcal{T})$ is a function of $\boldsymbol{\psi}_\text{meta}$ for a given task $\mathcal{T}$, making the expected validation loss an *implicit function* of $\boldsymbol{\psi}_\text{meta}$ alone. Assuming that we have done the inner optimization and obtained $\boldsymbol{\psi}^*_\text{adapt}(\boldsymbol{\psi}_\text{meta}, \mathcal{S}_\mathcal{T})$, in order to update the meta-learned parameters $\boldsymbol{\psi}_\text{meta}$, we need to compute the *total derivative* of the validation loss $\mathcal{L}_V$ w.r.t. $\boldsymbol{\psi}_\text{meta}$ for any given task $\mathcal{T}$:

$$\frac{d\,\mathcal{L}_V}{d\,\boldsymbol{\psi}_\text{meta}} = \frac{\partial\,\mathcal{L}_V}{\partial\,\boldsymbol{\psi}_\text{meta}} + \frac{\partial\,\mathcal{L}_V}{\partial\,\boldsymbol{\psi}^*_\text{adapt}} \frac{\partial\,\boldsymbol{\psi}^*_\text{adapt}}{\partial\,\boldsymbol{\psi}_\text{meta}}, \tag{3}$$

which is often referred to as *hypergradient*. In Equation (3), $\partial\,\mathcal{L}_V/\partial\,\boldsymbol{\psi}_\text{meta}$ and $\partial\,\mathcal{L}_V/\partial\,\boldsymbol{\psi}^*_\text{adapt}$ can be easily obtained by auto-differentiation. However, the computation of the derivative $\partial\,\boldsymbol{\psi}^*_\text{adapt}/\partial\,\boldsymbol{\psi}_\text{meta}$ of the best response function is more challenging, since $\boldsymbol{\psi}^*_\text{adapt}(\boldsymbol{\psi}_\text{meta}, \mathcal{S}_\mathcal{T})$ is defined by an $\arg\min$ function.

Algorithm 1 Exact hypergradient computation in ADKF-IFT.

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

Fortunately, Cauchy's *Implicit Function Theorem* (IFT) suggests a way of computing this quantity at the current meta-learned parameters ψ'_{meta} for a given task $\mathcal{T}'$:

$$\frac{\partial\, \psi^*_{\text{adapt}}}{\partial\, \psi_{\text{meta}}}\bigg|_{\psi'_{\text{meta}}} = -\left(\frac{\partial^2\, \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial\, \psi_{\text{adapt}}\, \partial\, \psi^T_{\text{adapt}}}\right)^{-1} \frac{\partial^2\, \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial\, \psi_{\text{adapt}}\, \partial\, \psi^T_{\text{meta}}}\Bigg|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}, \quad (4)$$

where $\psi'_{\text{adapt}} = \psi^*_{\text{adapt}}(\psi'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$. A full statement of the IFT in the context of ADKF-IFT can be found in Appendix A. The only potential problem with Equation (4) is the computation and inversion of the Hessian matrix for $\mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})$ w.r.t. ψ_{adapt}. This computation can be done exactly if $|A_{\Psi_{\text{adapt}}}|$ is small, which is the case in this paper (as will be discussed in Section 3.2). Otherwise, an approximation to the inverse Hessian (e.g., Neumann approximation [25, 6]) can be used, which reduces both the memory and computational complexities to $\mathcal{O}(|A_\Psi|)$. Combining (3) and (4), we have a recipe for computing the hypergradient ${d\,\mathcal{L}_V}/{d\,\psi_{\text{meta}}}$ exactly for a single task, as summarized in Algorithm 1. The expected hypergradient over $p(\mathcal{T})$ is approximated by averaging the hypergradients for a batch of K randomly sampled training tasks. The meta-learned parameters ψ_{meta} can then be updated with the averaged hypergradient using, e.g., the Adam optimizer [21]. This procedure is repeated until convergence. In practice, we evaluate the performance of our model on a small set of validation tasks during meta-training, and use early stopping [34] to avoid overfitting of ψ_{meta}.

At meta-test time, we make predictions for each unseen test task $\mathcal{T}_*$ based on the GP posterior predictive distribution with optimal parameters ψ^*_{meta} and $\psi^*_{\text{adapt}}(\psi^*_{\text{meta}}, \mathcal{S}_{\mathcal{T}_*})$:

$$p(\mathcal{Q}^y_{\mathcal{T}_*} \mid \mathcal{Q}^{\mathbf{x}}_{\mathcal{T}_*}, \mathcal{S}_{\mathcal{T}_*}, \psi^*_{\text{meta}}, \psi^*_{\text{adapt}}(\psi^*_{\text{meta}}, \mathcal{S}_{\mathcal{T}_*})). \quad (5)$$

3.2 Specific Implementation

In this paper, we consider the following specific implementation for the exact forms of the task-level loss functions $\mathcal{L}_T$ and $\mathcal{L}_V$ and the partition of A_Ψ. We give details for the case where a Gaussian likelihood with variance σ^2 is used, but this could be easily extended to Bernoulli/categorical likelihood [52, 40, 20]. We propose to use the negative log marginal likelihood $-\log p(\mathcal{S}^y_{\mathcal{T}} \mid \mathcal{S}^{\mathbf{x}}_{\mathcal{T}}, \psi_{\text{meta}}, \psi_{\text{adapt}})$ as the training loss $\mathcal{L}_T$, as is common practice:

$$\mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}) = \frac{1}{2}\left\langle \mathcal{S}^y_{\mathcal{T}}, \mathbf{K}^{-1}_{\mathcal{S}_{\mathcal{T}}}\, \mathcal{S}^y_{\mathcal{T}} \right\rangle + \frac{1}{2}\log\det(\mathbf{K}_{\mathcal{S}_{\mathcal{T}}}) + \frac{N_{\mathcal{S}_{\mathcal{T}}}}{2}\log(2\pi), \quad (6)$$

where $\mathbf{K}_{\mathcal{S}_{\mathcal{T}}} = k_{\psi_{\text{meta}}, \psi_{\text{adapt}}}(\mathcal{S}^{\mathbf{x}}_{\mathcal{T}}, \mathcal{S}^{\mathbf{x}}_{\mathcal{T}}) + \sigma^2\, \mathbf{I}_{N_{\mathcal{S}_{\mathcal{T}}}}$. We choose the validation loss $\mathcal{L}_V$ to be the negative log posterior predictive likelihood $-\log p(\mathcal{Q}^y_{\mathcal{T}} \mid \mathcal{Q}^{\mathbf{x}}_{\mathcal{T}}, \mathcal{S}_{\mathcal{T}}, \psi_{\text{meta}}, \psi_{\text{adapt}})$, also due to its common usage for making predictions with GPs:

$$\mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{T}) = -\log \mathcal{N}(\mathcal{Q}^y_{\mathcal{T}}; \mathbf{K}_{\mathcal{Q}_{\mathcal{T}}\, \mathcal{S}_{\mathcal{T}}}\, \mathbf{K}^{-1}_{\mathcal{S}_{\mathcal{T}}}\, \mathcal{S}^y_{\mathcal{T}}, \mathbf{K}_{\mathcal{Q}_{\mathcal{T}}} - \mathbf{K}_{\mathcal{Q}_{\mathcal{T}}\, \mathcal{S}_{\mathcal{T}}}\, \mathbf{K}^{-1}_{\mathcal{S}_{\mathcal{T}}}\, \mathbf{K}_{\mathcal{S}_{\mathcal{T}}\, \mathcal{Q}_{\mathcal{T}}}), \quad (7)$$

where $\mathbf{K}_{\mathcal{Q}_{\mathcal{T}}} = k_{\psi_{\text{meta}}, \psi_{\text{adapt}}}(\mathcal{Q}^{\mathbf{x}}_{\mathcal{T}}, \mathcal{Q}^{\mathbf{x}}_{\mathcal{T}}) + \sigma^2\, \mathbf{I}_{N_{\mathcal{Q}_{\mathcal{T}}}}$ and $\mathbf{K}_{\mathcal{S}_{\mathcal{T}}\, \mathcal{Q}_{\mathcal{T}}} = \mathbf{K}^T_{\mathcal{Q}_{\mathcal{T}}\, \mathcal{S}_{\mathcal{T}}} = k_{\psi_{\text{meta}}, \psi_{\text{adapt}}}(\mathcal{S}^{\mathbf{x}}_{\mathcal{T}}, \mathcal{Q}^{\mathbf{x}}_{\mathcal{T}})$. This has the advantage that the prediction procedure during meta-testing (defined in Equation (5)) exactly matches the meta-training procedure, thereby closely following the principle of *learning* *to learn*. As for the partition of A_Ψ, we propose to set $A_{\Psi_\text{adapt}} = A_\Theta$ and $A_{\Psi_\text{meta}} = A_\Phi$, i.e., to meta-learn the feature extractor parameters $\boldsymbol{\phi}$ across tasks and to adapt the GP hyperparameters $\boldsymbol{\theta}$ for each individual task. The reasons for this are as follows: 1) it has a clear and intuitive interpretation: we are 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; 2) since $|A_\Theta| < 10^2$ for essentially all common GP base kernels, the Hessian in Equation (4) can be computed and inverted exactly during meta-training using Algorithm 1; 3) the inner optimization (2) for $\boldsymbol{\psi}_\text{adapt}$ is computationally efficient, as it does not require backpropagating through the feature extractor $\mathbf{f}_{\boldsymbol{\phi}}$; and 4) we conjecture that adapting the GP hyperparameters 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.

4 Related Work

4.1 Learning Deep Kernels for Gaussian Processes

Many prior works have used neural network features to define kernels for GPs [17, 55, 54, 2, 3]. Disregarding differences in neural network architectures, optimization algorithms, and GP inferences, the training objective for nearly every prior work is to maximize either the marginal likelihood for a single dataset [17, 55, 3] or the expected marginal likelihood over a distribution of datasets [32]. These are often generically called Deep Kernel Learning (DKL) and Deep Kernel Transfer (DKT), respectively. Not only is our proposed ADKF-IFT method from Section 3.2 distinct from DKL and DKT, in fact both DKL and DKT can be viewed as *special cases* of our general ADKF-IFT framework from Section 3.1, with DKL being equivalent to $A_{\Psi_\text{meta}} = \varnothing$, $A_{\Psi_\text{adapt}} = A_\Psi$ and DKT being equivalent to $A_{\Psi_\text{meta}} = A_\Psi$, $A_{\Psi_\text{adapt}} = \varnothing$. By generalizing DKL and DKT, ADKF-IFT has the potential to avoid both the *overfitting* issue often seen in DKL [30] (by using meta-learning as a regularizer) and the *underfitting* issue in DKT [32] (by adapting some parameters to each task).

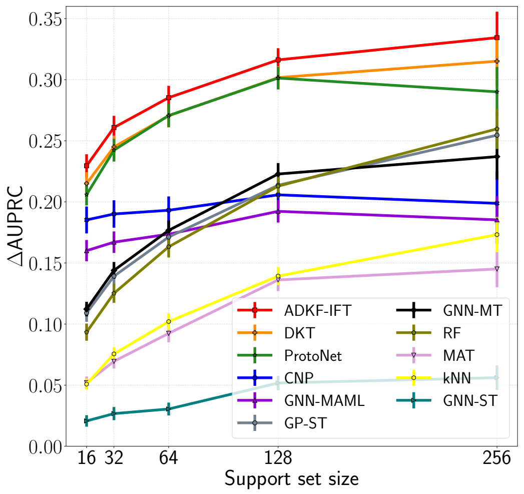
Adaptive Deep Kernel Learning-GP (ADKL-GP) is another method for adapting meta-learned GPs to specific tasks [46], which performs adaptation by embedding a support set using a meta-learned set encoder and inputting the resulting task embedding $\mathbf{z}_\mathcal{T}$ into the feature extractor. Their training objective is similar to that of DKT [32] but with an extra contrastive loss to regularize the set encoder. Unfortunately, this makes ADKL-GP very sensitive to the regularization coefficient γ, with improper tuning leading either to overfitting or to under-adaptation (being therefore equivalent to DKT). In contrast, our ADKF-IFT method requires no tuning and furthermore allows important parameters such as the noise level and lengthscales to be adapted, which almost certainly vary among datasets.

More broadly, our ADKF-IFT is particularly well-suited for molecular property prediction problems relative to other methods for the following reasons: 1) unlike for natural images, there are no pretrained generally useful feature extractors for molecules available[1], meaning that significant training of the feature extractor is required, which fully exploits the benefits of ADKF-IFT; 2) typical drug property prediction tasks have training sets of size $\sim 10^2$, which is large enough to adapt a small number of GP hyperparameters using ADKF-IFT but too small to use DKL without overfitting; and 3) there are a large number of small datasets available for different drug targets which share underlying physical mechanisms, making meta-training possible. However, ADKF-IFT may be less suitable in other settings where overfitting is less of a problem (e.g. image classification).

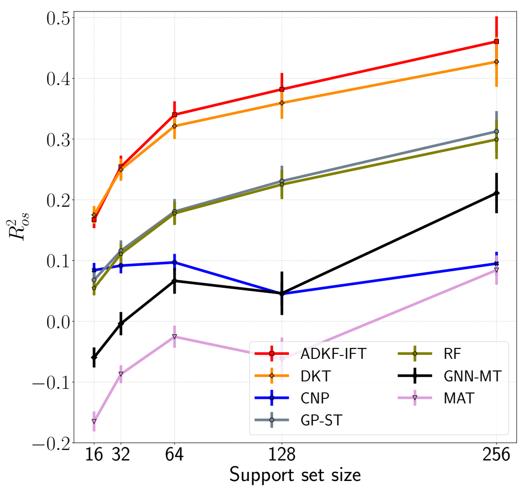
4.2 Other Work

Meta-learning together with pretraining has been used to achieve strong performance on few-shot learning problems in computer vision [22, 47, 49, 18, 31, 45, 4]. Some methods meta-learn a feature extractor which is used to parameterize a simpler predictor, similar to DKT. For example, ProtoNet [39] meta-learns a metric space for a nearest centroid classifier placed on top. CNP [13] constructs a conditional stochastic process by creating an embedding of a support set with a meta-learned set encoder. Like DKT, methods of this type tend to struggle with underfitting (see Figure 1 for empirical evidence). Other methods perform more significant test-time adaptation but constrain the adaptation

[1]There are a few reasons for this: 1) the total number of available measurements derived from wet-lab experiments is not comparable to the size of ImageNet [9]; and 2) unlike natural images, drug molecules lack meaningful invariances and local/global structures, and are not a clear manifold of the whole molecule space.



(a) Classification (157 tasks).



(b) Regression (111 tasks).

Figure 1: Mean performance with standard errors of all compared methods on all FS-Mol test tasks.

to avoid overfitting. The most popular method in this class is MAML [10] which fine-tunes a neural network while implicitly forcing the parameters to be close to a meta-learned initialization. Variants of MAML have also used implicit gradients to efficiently solve bilevel optimization problems [35, 5].

Finally, it should be noted that the implicit function theorem employed in our work has been used in many previous machine learning papers in various contexts, e.g., solving minimax optimization problems [50], tuning hyperparameters [1, 26, 33, 25, 6], and performing meta-learning [35, 23, 5].

5 Experiments

This section empirically demonstrates that the feature representations learned by ADKF-IFT from Section 3.2 are generally useful for both in-domain and out-of-domain tasks. As argued in Section 4.1, our method is expected to be especially well-suited for problems in drug discovery, and we therefore focus our experiments on problems in this domain. Specifically, we evaluate ADKF-IFT on the FS-Mol benchmark [41] (Section 5.1), and show that the ADKF-IFT feature representation is transferable to out-of-domain molecular property prediction and optimization tasks (Section 5.2).

We compare **ADKF-IFT** with four categories of learning algorithms: single-task methods, multi-task pretraining, self-supervised pretraining, and meta-learning methods. Representative methods are chosen within each category: Random Forest (**RF**), k-Nearest Neighbors (**kNN**), single-task GP with Tanimoto kernel (**GP-ST**) [36], single-task GNN (**GNN-ST**) [15], multi-task GNN (**GNN-MT**) [8, 15], Molecule Attention Transformer (**MAT**) [27], Prototypical Network with Mahalanobis distance (**ProtoNet**) [39], Model-Agnostic Meta-Learning (**GNN-MAML**) [10], Conditional Neural Process (**CNP**) [13], and Deep Kernel Transfer (**DKT**) [32]. We chose not to compare against ADKL-GP [46] due to difficulties tuning its regularization hyperparameters (see Appendix D.3). The feature extractor architecture $\mathbf{f}_\phi$ used for CNP, DKT, and ADKF-IFT is the same as that used for ProtoNet in [41]. We use Matérn52 without automatic relevance determination (ARD) [29] as the base kernel in DKT and 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. For ADKF-IFT, the lengthscale is initialized using the median heuristic [14] for each task, with a log-normal prior centered at the initialization. Detailed configurations of all compared methods can be found in Appendix B.

5.1 Few-shot Molecular Property Prediction

Benchmark. We conduct our evaluation on the FS-Mol benchmark [41], 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 [28], split into training (4,938 tasks), validation (40 tasks), and test (157 tasks) sets. Each task is associated with a protein target. The

Table 1: 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	9.59	9.81	10.02	10.16	10.49
kNN	9.17	8.89	8.75	8.75	8.23
MAT	8.81	8.89	8.73	8.43	8.49
RF	6.88	6.74	6.21	5.68	4.42
GNN-MT	6.36	6.27	6.28	6.00	6.21
GP-ST	5.79	5.75	5.67	5.64	5.09
GNN-MAML	5.72	6.20	6.64	7.12	8.23
CNP	4.49	5.20	5.71	6.25	7.02
ProtoNet	3.72	3.21	3.00	2.88	3.76
DKT	3.22	3.04	2.91	2.91	2.67
ADKF-IFT	**2.25**	**2.00**	**2.07**	**2.18**	**1.38**

(b) Regression (111 tasks).

Method	Support set size				
	16	32	64	128	256
MAT	6.60	6.54	6.48	6.38	6.51
GNN-MT	5.72	5.67	5.51	5.50	5.23
RF	4.38	4.11	3.94	3.55	3.53
GP-ST	3.70	3.77	3.66	3.32	3.07
CNP	3.54	4.09	4.55	5.27	5.86
DKT	2.05	2.01	2.22	2.27	2.43
ADKF-IFT	**2.01**	**1.82**	**1.64**	**1.72**	**1.36**

original benchmark only considers binary classification of active/inactive compounds, since regressing the actual numeric activity target (IC50 or EC50) is known to be extremely difficult due to, e.g., large measurement noise. We nevertheless include regression (for the log numeric activity target) in our evaluation, as it is a desired and more preferred task to do in real-world drug discovery projects. 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 [42]. Following [32], we treat binary classification as ± 1 label regression in GP-ST, DKT, and ADKF-IFT. The classification results for RF, kNN, GNN-ST, GNN-MT, MAT, ProtoNet, and GNN-MAML are taken from [41].

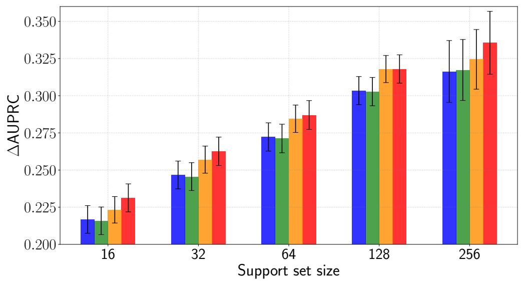
Evaluation Procedure. The task-level metrics for binary classification and regression are ΔAUPRC (change in area under the precision-recall curve) and R^2_{os} (predictive/out-of-sample coefficient of determination), respectively. Details of these metrics can be found in Appendix C. We follow exactly the same evaluation procedure as that in [41], where the averaged performance over ten different stratified support/query random splits of every test task is reported for each compared method. This evaluation process is performed for five different support set sizes 16, 32, 64, 128, and 256.

Overall Performance. Figure 1 shows the overall test performance of all compared methods. Note that RF is considered a strong baseline method, since it is widely used in real-world drug discovery projects and has comparable performance to multi-task and self-supervised 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 Table 1, we show 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 1. In Appendix D.1, we show box plots of the overall test performance of all compared methods.

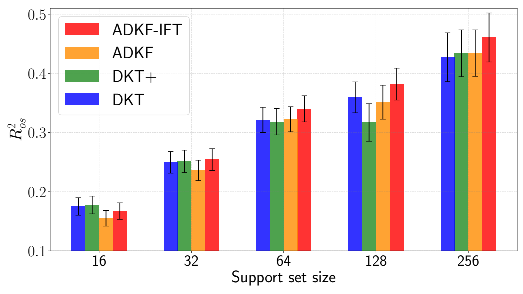
Statistical Comparison. We perform two-sided Wilcoxon signed-rank tests [53] to compare the performance of ADKF-IFT and the next best method, namely DKT. The exact p-values from these statistical tests can be found in Appendix D.2. The results indicate that ADKF-IFT *significantly* outperforms DKT for 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 in learning adaptive feature representations and 2) the performance gains of ADKF-IFT do not just come from tuning the GP hyperparameters $\boldsymbol{\theta}$ at meta-test time, we consider two ablation models: DKT+ and ADKF. The test performance of these models are shown in Figure 2. 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 adjust the GP hyperparameters $\boldsymbol{\theta}$ for each task at meta-test time. The results show that DKT+ has similar or worse performance compared to DKT, indicating that tuning the GP hyperparameters $\boldsymbol{\theta}$ 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 (EC) number [51], which enables sub-benchmark evaluation within the entire



(a) Classification (157 tasks).



(b) Regression (111 tasks).

Figure 2: 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 GP hyperparameters $\boldsymbol{\theta}$ during meta-testing.

Table 2: Mean performance with standard errors of selected 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
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 ± 0.028	$\mathbf{0.231 \pm 0.022}$
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	$\mathbf{0.285 \pm 0.010}$

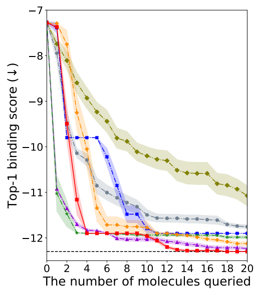
(b) Regression (R^2_{os}).

FS-Mol sub-benchmark (EC category)			Method				
Class	Description	#tasks	RF	GP-ST	CNP	DKT	ADKF
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	$\mathbf{0.340 \pm 0.022}$

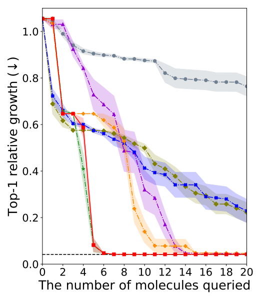
benchmark. Ideally, the best method should be able to perform well across all sub-benchmarks. Table 2 shows the test performance of top performing methods on all sub-benchmarks at support set size 64 (the median of all considered support sizes) for both the classification and regression tasks. The results 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.

5.2 Out-of-domain Molecular Property Prediction and Optimization

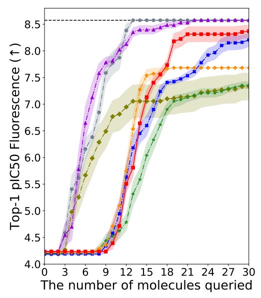
We further demonstrate that the feature representation learned by ADKF-IFT is useful not only for in-domain molecular property prediction tasks but also for out-of-domain molecular property prediction and optimization tasks. For this, we perform experiments involving finding molecules with best desired target properties within given datasets using Bayesian optimization (BO) with a GP surrogate model operating on top of compared feature representations. We use the expected improvement acquisition function [19] with query-batch size 1. All compared feature representations are extracted from the models trained on the FS-Mol dataset from scratch in Section 5.1, except



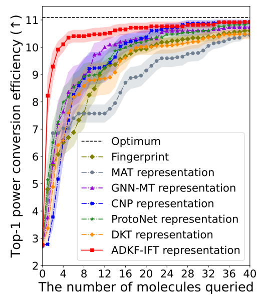
(a) Molecular docking.



(b) Antibiotic discovery.



(c) Antiviral drug design.



(d) Material design.

Figure 3: 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 3: 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
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	$\mathbf{0.996 \pm 0.007}$

for the pretrained MAT representation and fingerprint. We compare them on four representative molecular design tasks. Detailed configuration of the GP and descriptions of the tasks can be found in Appendix E. We repeat each BO experiment 20 times, each time starting from 16 randomly sampled molecules from the worst $\sim$ 700 molecules within the dataset. Results of these experiments are shown in Figure 3. It can be seen 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.

In addition, in Table 3 we explicitly report the regression predictive performance of a GP operating on top of each compared feature representation for the 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 achieves competitive performance on the antiviral drug design task.

6 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 unified 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 prediction tasks and on out-of-domain molecular property prediction and optimization tasks.

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 [5]); 3) adopting a more principled approximate inference strategy for few-shot GP classification (e.g., Pólya-Gamma data augmentation [40] or Laplace approximation [20]); and 4) injecting domain expertise in drug discovery into the base kernel with hand-curated features and kernel combinations.

Acknowledgments and Disclosure of Funding

We thank Massimiliano Patacchiola, John Bronskill, Marcin Sendera, and Richard E. Turner for helpful discussions and feedback. WC acknowledges funding via a Cambridge Trust Scholarship (supported by the Cambridge Trust) and a Cambridge University Engineering Department Studentship (under grant G105682 NMZR/089 supported by Huawei R&D UK). AT acknowledges funding via a C T Taylor Cambridge International Scholarship. JMHL acknowledges support from a Turing AI Fellowship under grant EP/V023756/1.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[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] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

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

[37] 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.

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

[39] 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.

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

[41] 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.

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

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.

[48] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

[49] 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.

[50] Yuanhao Wang*, Guodong Zhang*, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *International Conference on Learning Representations*, 2020.

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

[52] Florian Wenzel, Théo Galy-Fajou, Christan Donner, Marius Kloft, and Manfred Opper. Efficient gaussian process classification using pólya-gamma data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5417–5424, 2019.

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

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

[55] 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.

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 ψ'_{meta} and ψ'_{adapt} that $\left.\frac{\partial \mathcal{L}_\mathcal{T}(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}}\right|_{\psi'_{meta},\psi'_{adapt}} = \mathbf{0}$. Suppose that $\frac{\partial \mathcal{L}_\mathcal{T}}{\partial \psi_{adapt}}(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'}) : \Psi_{meta} \times \Psi_{adapt} \to \Psi_{adapt}$ is a continuously differentiable function w.r.t. ψ_{meta} and ψ_{adapt}, and the Hessian $\left.\frac{\partial^2 \mathcal{L}_\mathcal{T}(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}\,\partial \psi_{adapt}^T}\right|_{\psi'_{meta},\psi'_{adapt}}$ is invertible. Then, there exists an open set $U \in \Psi_{meta}$ containing ψ'_{meta} and a function $\psi^*_{adapt}(\psi_{meta},\mathcal{S}_{\mathcal{T}'}) : \Psi_{meta} \to \Psi_{adapt}$, such that $\psi'_{adapt} = \psi^*_{adapt}(\psi'_{meta},\mathcal{S}_{\mathcal{T}'})$ and $\left.\frac{\partial \mathcal{L}_\mathcal{T}(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}}\right|_{\psi''_{meta},\psi^*_{adapt}(\psi''_{meta},\mathcal{S}_{\mathcal{T}'})} = \mathbf{0}$, $\forall \psi''_{meta} \in U$. Moreover, the rate at which $\psi^*_{adapt}(\psi_{meta},\mathcal{S}_{\mathcal{T}'})$ is changing w.r.t. ψ_{meta} for any $\psi''_{meta} \in U$ is given by*

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

B Detailed Configurations of All Compared Methods for FS-Mol

Single-task Methods. Single-task methods (RF, kNN, GP-ST, and GNN-ST) 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 [41]. RF, kNN, and GP-ST operates on top of manually curated features obtained using RDKit. RF and kNN use extended connectivity fingerprint [38] (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). Hyperparameter search configurations for these methods are based on the extensive industrial experience from the authors of [41]. GNN-ST uses a 8-layer GNN with a hidden dimension of 128 and a gated readout function [15], considering $\sim$ 30 hyperparameter search configurations.

Multi-task Pretraining. The implementation of GNN-MT is taken from [41]. GNN-MT shares a 10-layer GNN with a hidden dimension of 128 using principal neighborhood message aggregation [8] 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 [41]. We use the official pretrained model parameters [27], which is pretrained on 2 millions molecules sampled from the ZINC15 dataset [42]. We fine-tuned it for each test task with hyperparameter search and early stopping based on 20% of the support set associated with the task.

Meta-learning Methods. Meta-learning methods (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 [41]. The GNN feature extractor used for GNN-MAML is the same as that used for GNN-ST. ProtoNet, CNP, DKT, and ADKF-IFT operates on top of a combination of extended connectivity fingerprint [38] (count-based fingerprint with radius 2 and size 2,048) and features extracted by a GNN. The feature extractor architecture used for CNP, DKT, and ADKF-IFT is the same as that used for ProtoNet in [41], with the size of the feature representation being tuned on the validation tasks. We use Matérn52 without automatic relevance determination (ARD) [29] as the base kernel in DKT and 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. For ADKF-IFT, the lengthscale is initialized using the median heuristic [14] with a log-normal prior centered at the initialization.

Table 4: 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 5: Descriptions of four out-of-domain molecular design tasks.

Molecular design task	Data source	#compounds	Target	Target source
Molecular docking (ESR2)	DOCKSTRING training set [12]	2,312	binding score	AutoDock Vina [48]
Antibiotic discovery (E. coli BW25113)	Antibiotic training set [43]	2,335	relative growth	screening
Antiviral drug design (SARS-CoV-2)	COVID Moonshot [7]	1,926	pIC50 Fluorescence	experimental lab
Material design (Organic Photovoltaic)	Harvard Clean Energy Project [16]	2,012	power conversion efficiency	DFT simulation

C Task-level Evaluation Metrics for FS-Mol

Binary Classification. Following [41], 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 $\mathcal{Q}_\mathcal{T}$.

D Further Experimental Results on FS-Mol

D.1 Overall Performance in Box Plots

Figures 5 and 6 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 1.

D.2 Statistical Comparison

Table 4 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 $\alpha = 0.05$).

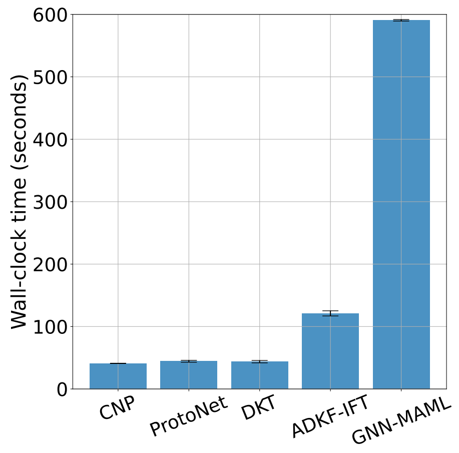


Figure 4: Wall-clock time consumed (with standard errors) when meta-testing on a pre-defined set of FS-Mol classification tasks using each of the compared meta-learning methods.

D.3 Comments on Experiments with ADKL-GP

As stated in section 4.1, the training objective for ADKL-GP [46] is equivalent to the objective for DKT with an added contrastive loss, weighted by a factor γ (see Equation (13) in their preprint [46]). 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 [55].
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 [32].

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, the paper [46] 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, and felt that it would be unfair to report only preliminary results that were not checked as thoroughly as the other baselines.

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, as mentioned in Section 4.1, our ADKF-IFT is able to adapt important parameters such as the likelihood noise level and kernel lengthscales, 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 5.

D.4 Meta-testing Costs

Figure 4 shows the meta-testing costs of all compared meta-learning methods in terms of wall-clock time[2] 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,

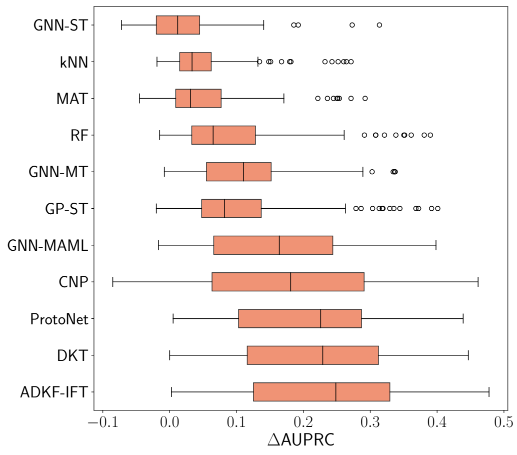
[2]We acknowledge that wall-clock time may not be the best metric for measuring the costs, since some meta-learning methods could be parallelized, which will reduce the wall-clock time accordingly. An alternative metric is multiply–accumulate operation (MAC). However, it is difficult to obtain the accurate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

and DKT, but still much faster than GNN-MAML. 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.

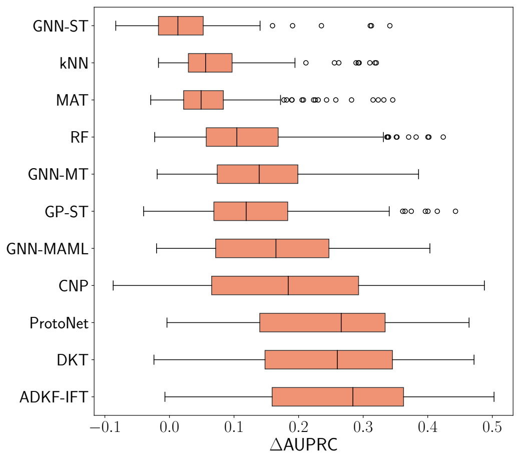
E Details of the Out-of-domain Molecular Optimization Experiments

In Table 5, we summarize the four molecular design tasks considered in Section 5.2. 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 [16], respectively.

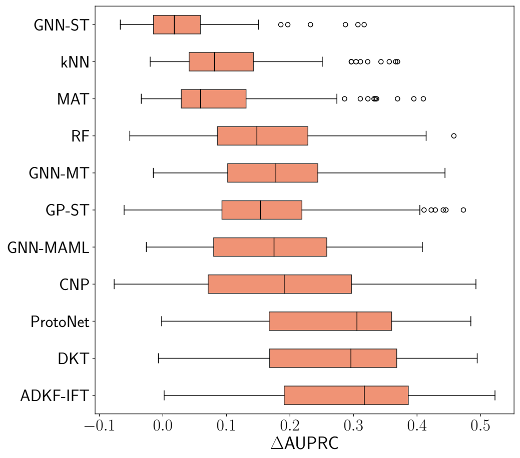
For the configuration of the GP, we use the Tanimoto kernel for fingerprint (with radius 2 and 2,048 bits based on count simulation) and Matérn52 kernel without ARD (with a log-normal prior over the lengthscale, centered at the median heuristic initialization) for all the other compared feature representations.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



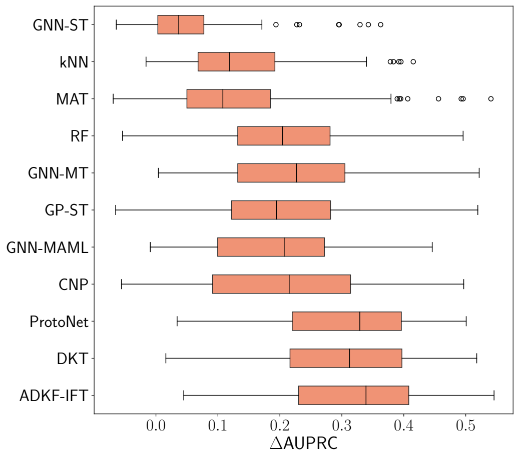
(a) $N_{\mathcal{S}_{\mathcal{T}_*}} = 16$.



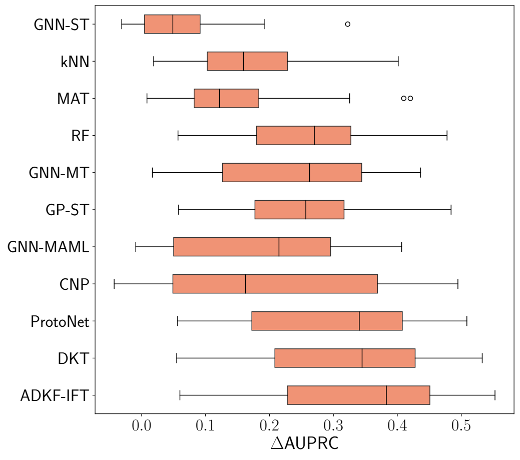
(b) $N_{\mathcal{S}_{\mathcal{T}_*}} = 32$.



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

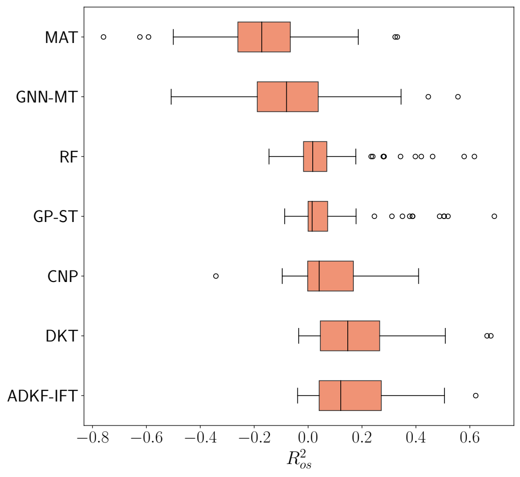


(d) $N_{\mathcal{S}_{\mathcal{T}_*}} = 128$.

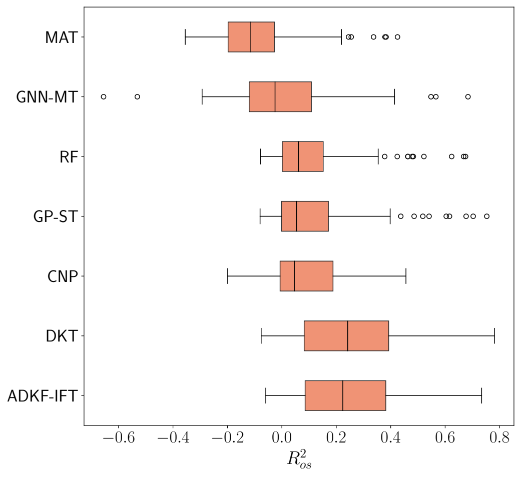


(e) $N_{\mathcal{S}_{\mathcal{T}_*}} = 256$.

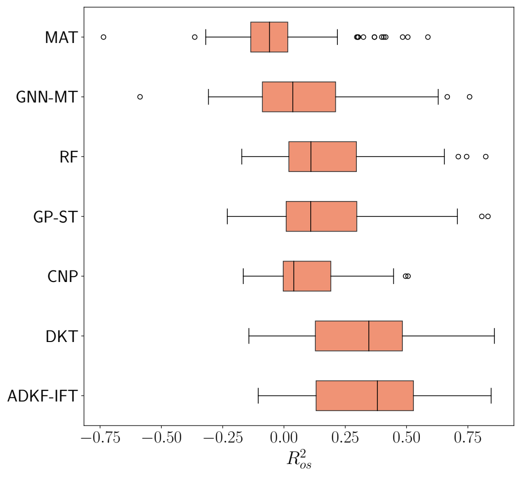
Figure 5: Box plots for the classification performance of all compared methods on 157 FS-Mol test tasks at different support set sizes.



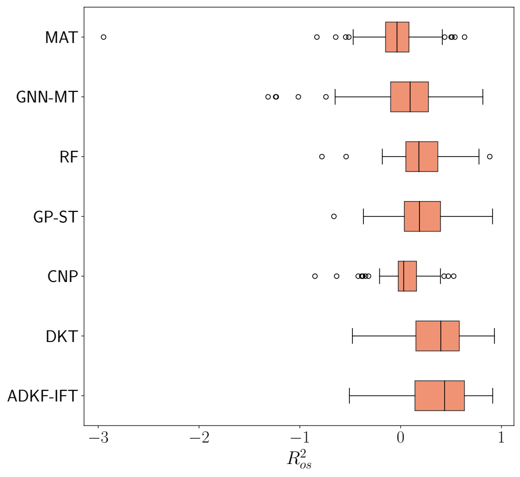
(a) $N_{\mathcal{S}_{\mathcal{T}_*}} = 16$.



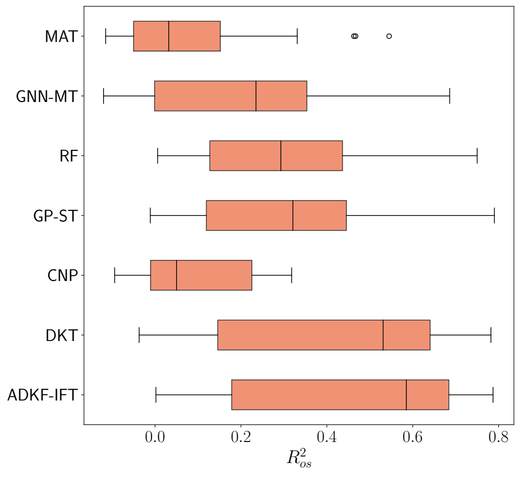
(b) $N_{\mathcal{S}_{\mathcal{T}_*}} = 32$.



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



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



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

Figure 6: Box plots for the regression performance of all compared methods on 111 FS-Mol test tasks at different support set sizes.