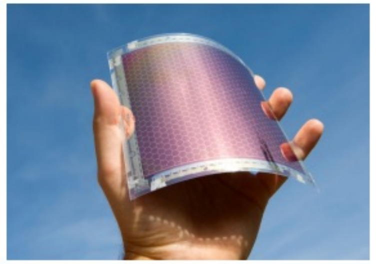
Meta-learning Adaptive Deep Kernel Gaussian Processes for Molecular Property Prediction

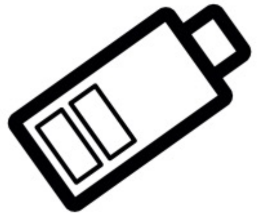
Wenlin Chen   Austin Tripp   José Miguel Hernández-Lobato

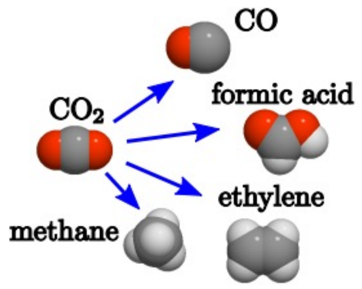
Published at ICLR 23: https://openreview.net/forum?id=KXRSh0sdVTP

Motivation: Molecular Design

Goal: find novel molecules with desirable biochemical/physicochemical properties.

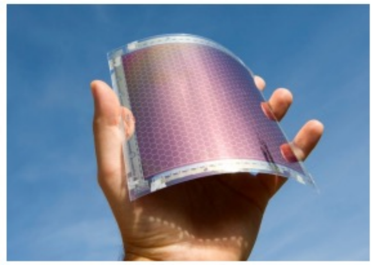
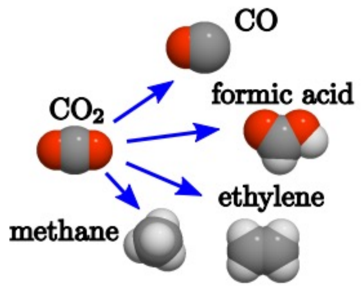






Motivation: Molecular Design

Goal: find novel molecules with desirable biochemical/physicochemical properties.

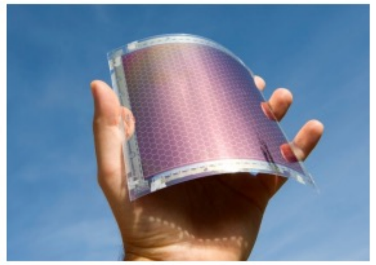
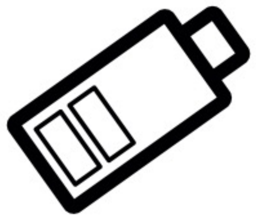
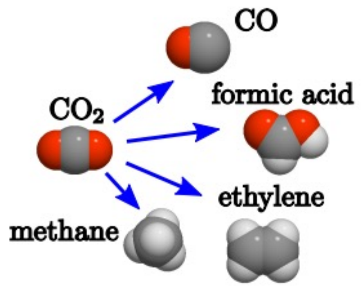



Challenge:

- Evaluating molecular properties is slow and expensive: small datasets are ubiquitous.
- Chemical space is huge and complex: exhaustive search is prohibitive.

Motivation: Molecular Design

Goal: find novel molecules with desirable biochemical/physicochemical properties.

Challenge:

- Evaluating molecular properties is slow and expensive: small datasets are ubiquitous.
- Chemical space is huge and complex: exhaustive search is prohibitive.

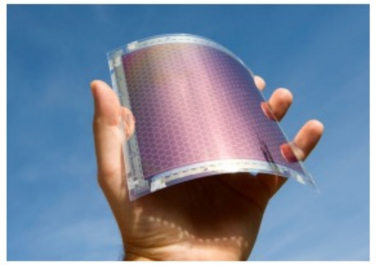
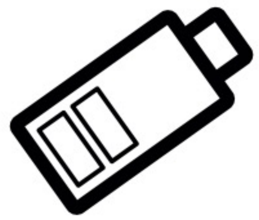
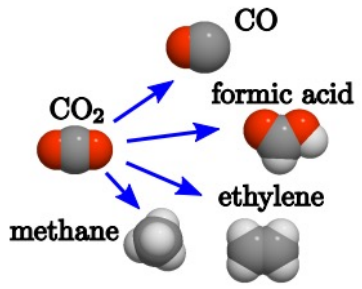
Gaussian processes (GPs):

✓ GPs are well-calibrated models with generally reliable uncertainty on small datasets.

✓ GPs could be used as surrogate models in Bayesian optimization to guide molecule search.

Motivation: Molecular Design

Goal: find novel molecules with desirable biochemical/physicochemical properties.

Challenge:

- Evaluating molecular properties is slow and expensive: small datasets are ubiquitous.
- Chemical space is huge and complex: exhaustive search is prohibitive.

Gaussian processes (GPs):

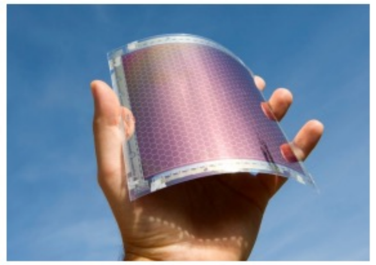
✓ GPs are well-calibrated models with generally reliable uncertainty on small datasets.

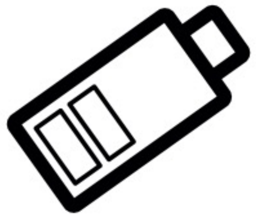
✓ GPs could be used as surrogate models in Bayesian optimization to guide molecule search.

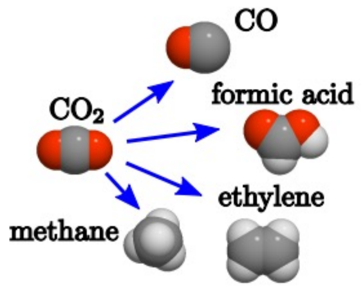
✗ Hand-designing kernels for structured data like molecules is challenging.

Motivation: Molecular Design

Goal: find novel molecules with desirable biochemical/physicochemical properties.







Challenge:

- Evaluating molecular properties is slow and expensive: small datasets are ubiquitous.

- Chemical space is huge and complex: exhaustive search is prohibitive.

Gaussian processes (GPs):

✓ GPs are well-calibrated models with generally reliable uncertainty on small datasets.

✓ GPs could be used as surrogate models in Bayesian optimization to guide molecule search.

× Hand-designing kernels for structured data like molecules is challenging.

We want better GP models for molecules!

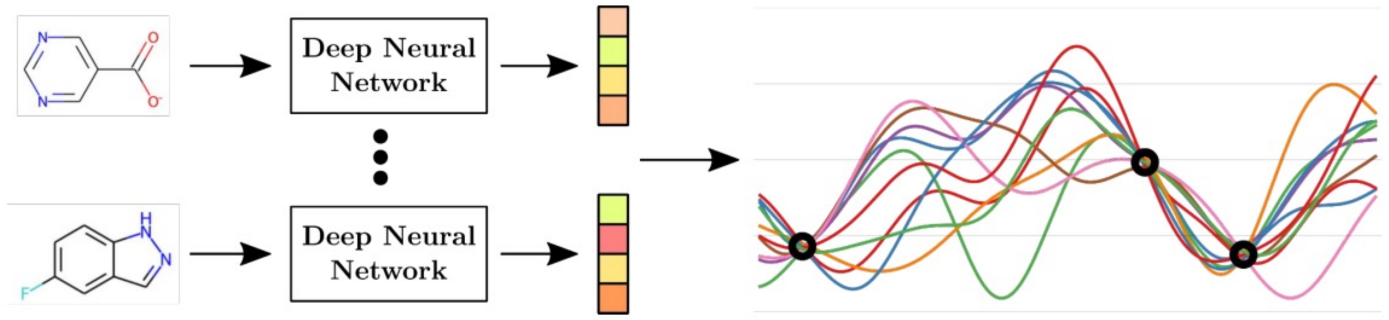
Model: Deep Kernel Gaussian Processes

Why not just learn features for molecules using a deep neural network (DNN)?

Model: Deep Kernel Gaussian Processes

Why not just learn features for molecules using a deep neural network (DNN)?

Deep kernel GPs operate on features learned by a DNN.

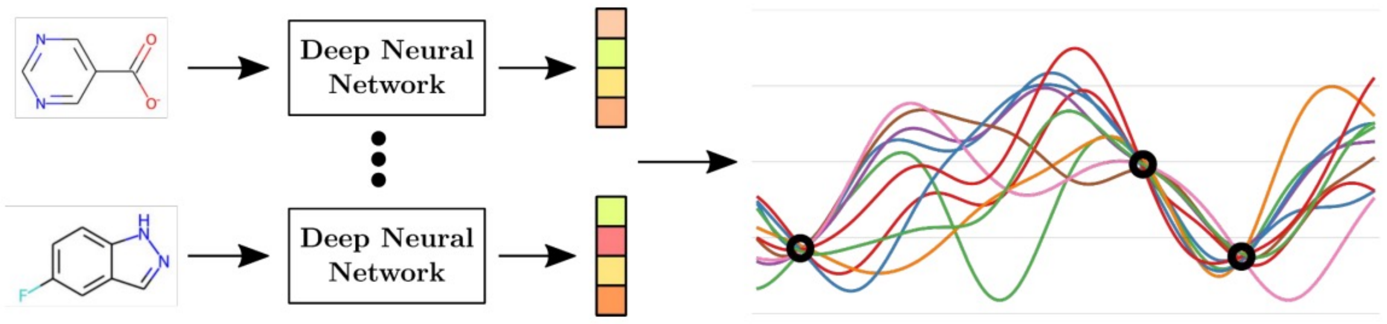


Representation learning (DNN) + Uncertainty (GP)

Model: Deep Kernel Gaussian Processes

Why not just learn features for molecules using a deep neural network (DNN)?

Deep kernel GPs operate on features learned by a DNN.



Representation learning (DNN) + Uncertainty (GP)

$$k_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{x}') = c_{\boldsymbol{\theta}}(\mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{x}), \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{x}'))$$

deep kernel
learnable parameters $\psi = (\boldsymbol{\theta}, \boldsymbol{\phi})$

base kernel
(e.g., RBF)

DNN feature
extractor

Previous Methods for Training Deep Kernel GPs

1. Deep Kernel Learning (**DKL**, Wilson et al., 2016)

- All parameters are learned by minimizing the negative log marginal likelihood (NLML) **on a single dataset**.

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \mathrm{NLML}\left(\boldsymbol{\psi}, \mathcal{S}_{\mathcal{T}}\right)$$

- Pure **single-task learning** (a separate deep kernel GP is trained for each task).

Wilson, Andrew Gordon, et al. "Deep kernel learning." Artificial intelligence and statistics. PMLR, 2016.

Ober, Sebastian W., Carl E. Rasmussen, and Mark van der Wilk. "The promises and pitfalls of deep kernel learning." Uncertainty in Artificial Intelligence. PMLR, 2021.

Previous Methods for Training Deep Kernel GPs

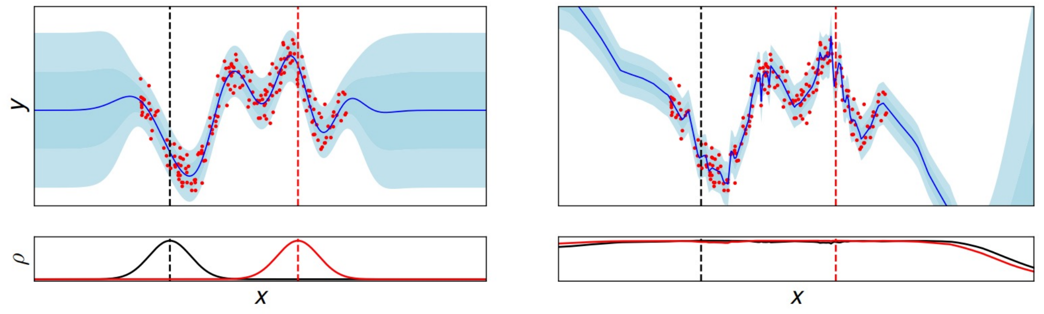
1. Deep Kernel Learning (**DKL**, Wilson et al., 2016)

- All parameters are learned by minimizing the negative log marginal likelihood (NLML) **on a single dataset**.

$$\psi^* = \arg\min_{\psi} \mathrm{NLML}\left(\psi, \mathcal{S}_{\mathcal{T}}\right)$$

- Pure **single-task learning** (a separate deep kernel GP is trained for each task).

× **Severe overfitting** (Ober et al., 2021) on **small datasets** despite the use of type-II ML.



(a) SE kernel (b) Exact DKL kernel

DKL makes all output values strongly correlated!

Wilson, Andrew Gordon, et al. "Deep kernel learning." Artificial intelligence and statistics. PMLR, 2016.

Ober, Sebastian W., Carl E. Rasmussen, and Mark van der Wilk. "The promises and pitfalls of deep kernel learning." Uncertainty in Artificial Intelligence. PMLR, 2021.

Previous Methods for Training Deep Kernel GPs

2. Deep Kernel Transfer (**DKT**, Patacchiola et al., 2020)

- All parameters are learned by minimizing the expected NLML **over a distribution of datasets**.

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \mathbb{E}_{p(\mathcal{T})}[\mathrm{NLML}(\boldsymbol{\psi}, \mathcal{T})]$$

- Pure **meta-learning** (all parameters are shared across tasks).

Patacchiola, Massimiliano, et al. "Bayesian meta-learning for the few-shot setting via deep kernels." Advances in Neural Information Processing Systems 33 (2020): 16108-16118.

Previous Methods for Training Deep Kernel GPs

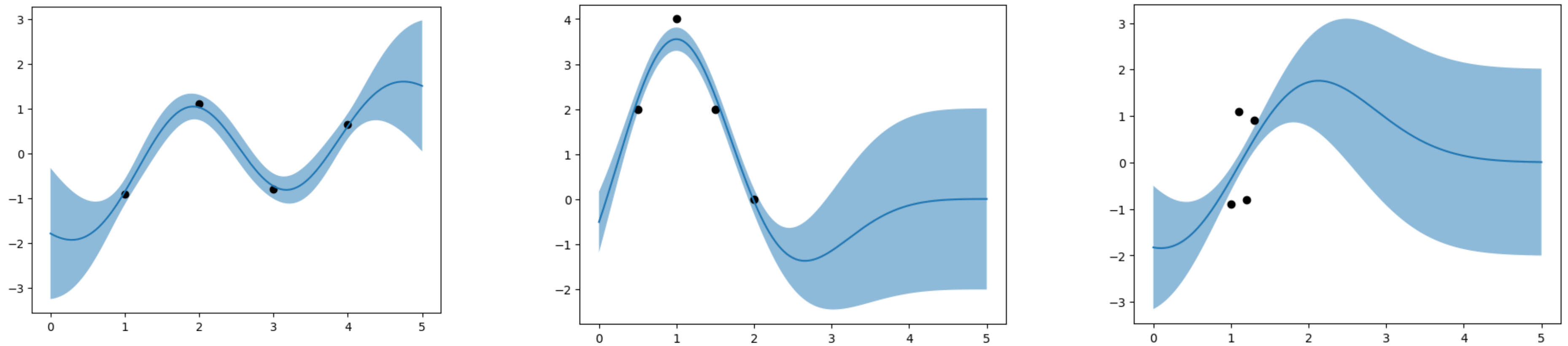
2. Deep Kernel Transfer (**DKT**, Patacchiola et al., 2020)

- All parameters are learned by minimizing the expected NLML **over a distribution of datasets**.

$$\psi^* = \arg\min_{\psi} \mathbb{E}_{p(\mathcal{T})}[\mathrm{NLML}(\psi, \mathcal{T})]$$

- Pure **meta-learning** (all parameters are shared across tasks).

✗ **Underfitting** due to **model mis-specification**.



It's unrealistic to assume all datasets are drawn from an identical GP with the same noise, signal variance, characteristic lengthscales!

Patacchiola, Massimiliano, et al. "Bayesian meta-learning for the few-shot setting via deep kernels." Advances in Neural Information Processing Systems 33 (2020): 16108-16118.

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

- **ADKF-IFT** interpolates between DKL and DKT:

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

- **ADKF-IFT** interpolates between DKL and DKT:
    - Partition the deep kernel parameters $\boldsymbol{\psi} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$ into two disjoint sets $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ and $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$.

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

- **ADKF-IFT** interpolates between DKL and DKT:
  - Partition the deep kernel parameters $\boldsymbol{\psi} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$ into two disjoint sets $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ and $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$.
  - Adapt base kernel parameters $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ to each task's training set $\mathcal{S}_{\mathcal{T}}$ by minimizing the NLML $\mathcal{L}_{\mathcal{T}}$ train loss.

$$\psi^{*}_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}) = \underset{\psi_{\text{adapt}}}{\arg\min} \ \mathcal{L}_{\mathcal{T}}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}). \longleftarrow$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

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

- **ADKF-IFT** interpolates between DKL and DKT:

  - Partition the deep kernel parameters $\boldsymbol{\psi} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$ into two disjoint sets $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ and $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$.

  - Adapt base kernel parameters $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ to each task's training set $\mathcal{S}_{\mathcal{T}}$ by minimizing the NLML $\mathcal{L}_{\mathcal{T}}$ train loss.

  - Meta-learn feature extractor parameters $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ to optimize the model's average performance on the test sets $\mathcal{Q}_{\mathcal{T}}$ of many tasks (after $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ has been separately adapted to each of these tasks).

$$\psi_{\text{meta}}^* = \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}^*(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}), \mathcal{T})],$$

$$\text{s.t.} \quad \psi_{\text{adapt}}^*(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}) = \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_{\mathcal{T}}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}). \longleftarrow$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

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

- ADKF-IFT interpolates between DKL and DKT:
  - Partition the deep kernel parameters $\boldsymbol{\psi} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$ into two disjoint sets $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ and $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$.
  - Adapt base kernel parameters $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ to each task's training set $\mathcal{S}_\mathcal{T}$ by minimizing the NLML $\mathcal{L}_\mathcal{T}$ train loss.
  - Meta-learn feature extractor parameters $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ to optimize the model's average performance on the test sets $\mathcal{Q}_\mathcal{T}$ of many tasks (after $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ has been separately adapted to each of these tasks).
- The validation loss $\mathcal{L}_V$ is the negative log joint predictive posterior on the test set $\mathcal{Q}_\mathcal{T}$ given the training set $\mathcal{S}_\mathcal{T}$.

$$\psi^*_{\text{meta}} = \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})],$$

$$\text{s.t.}\quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_\mathcal{T}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}). \longleftarrow$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

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

- ADKF-IFT interpolates between DKL and DKT:

  - Partition the deep kernel parameters $\boldsymbol{\psi} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$ into two disjoint sets $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ and $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$.

  - Adapt base kernel parameters $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ to each task's training set $\mathcal{S}_{\mathcal{T}}$ by minimizing the NLML $\mathcal{L}_{\mathcal{T}}$ train loss.

  - Meta-learn feature extractor parameters $\boldsymbol{\psi}_{meta} = \boldsymbol{\phi}$ to optimize the model's average performance on the test sets $\mathcal{Q}_{\mathcal{T}}$ of many tasks (after $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ has been separately adapted to each of these tasks).

- The validation loss $\mathcal{L}_V$ is the negative log joint predictive posterior on the test set $\mathcal{Q}_{\mathcal{T}}$ given the training set $\mathcal{S}_{\mathcal{T}}$.

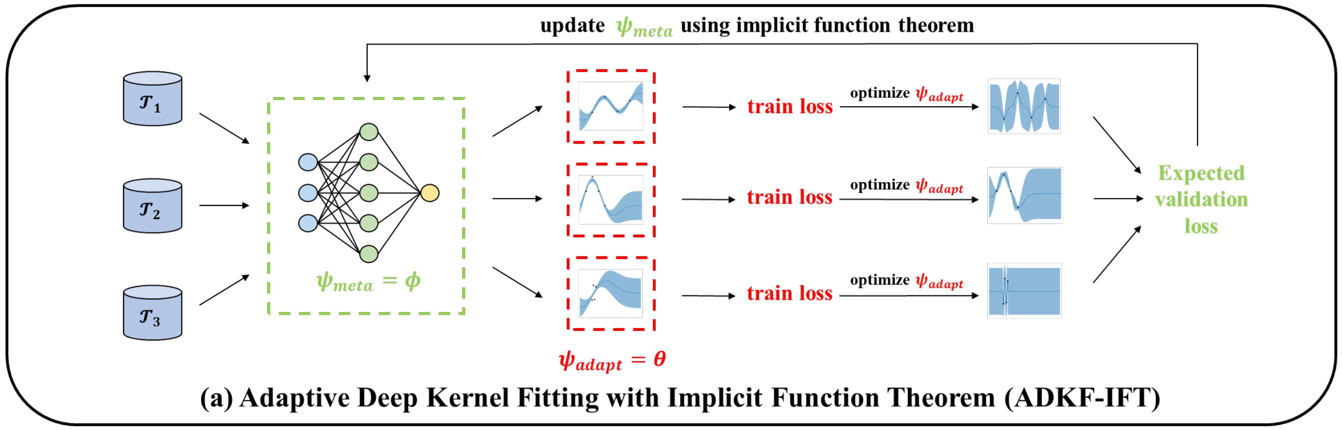
- **Meta-training**: ADKF-IFT can be formalized as a bi-level optimization problem.

$$\begin{aligned} \psi^*_{\text{meta}} &= \operatorname*{arg\,min}_{\psi_{\text{meta}}} \; \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}), \mathcal{T})], \\ \text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}) &= \operatorname*{arg\,min}_{\psi_{\text{adapt}}} \; \mathcal{L}_{\mathcal{T}}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}). \longleftarrow \end{aligned}$$

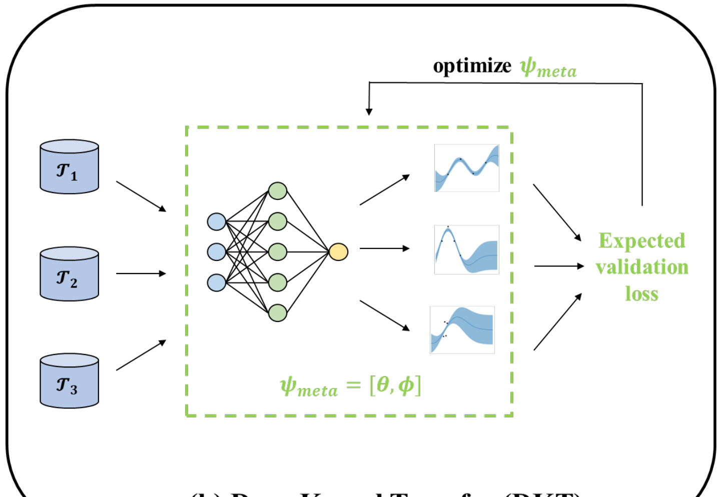
best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Interpretation:** DNN meta-learns generally useful features across tasks, such that a task-specific GP operates on top of such features achieves the highest predictive performance on average.

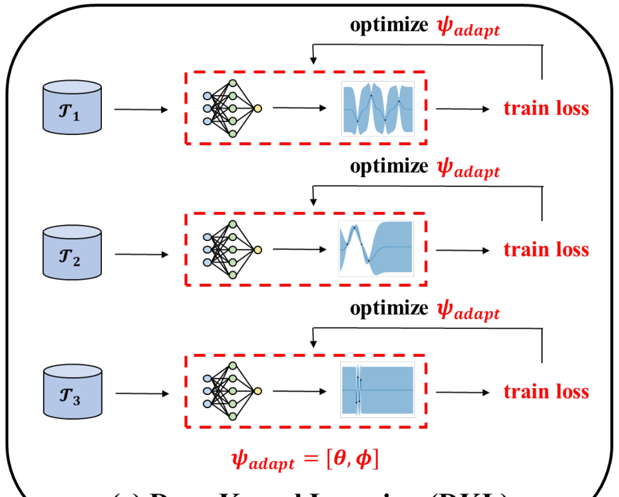
Contrast DKL, DKT and ADKF-IFT



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

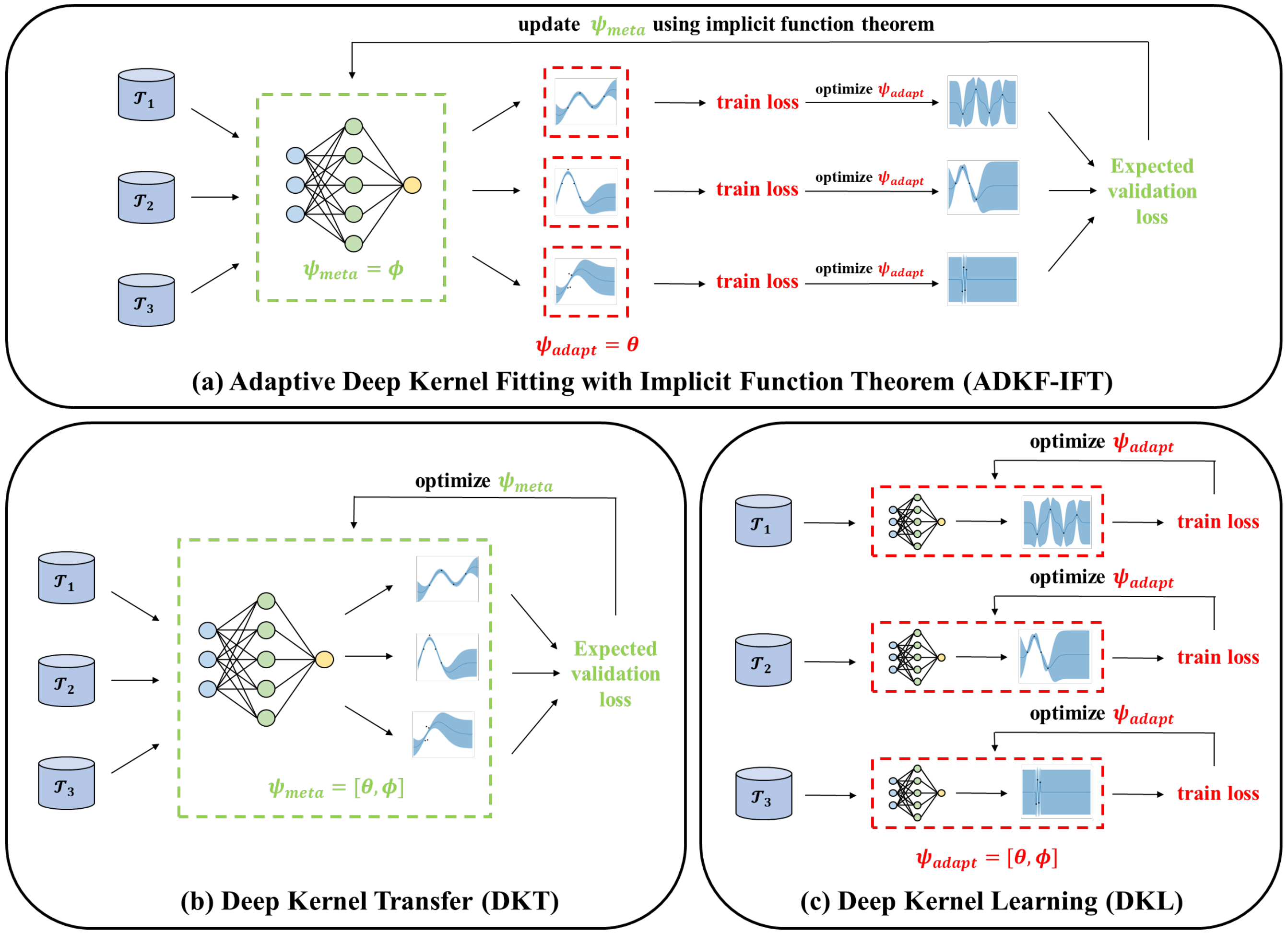


(b) Deep Kernel Transfer (DKT)



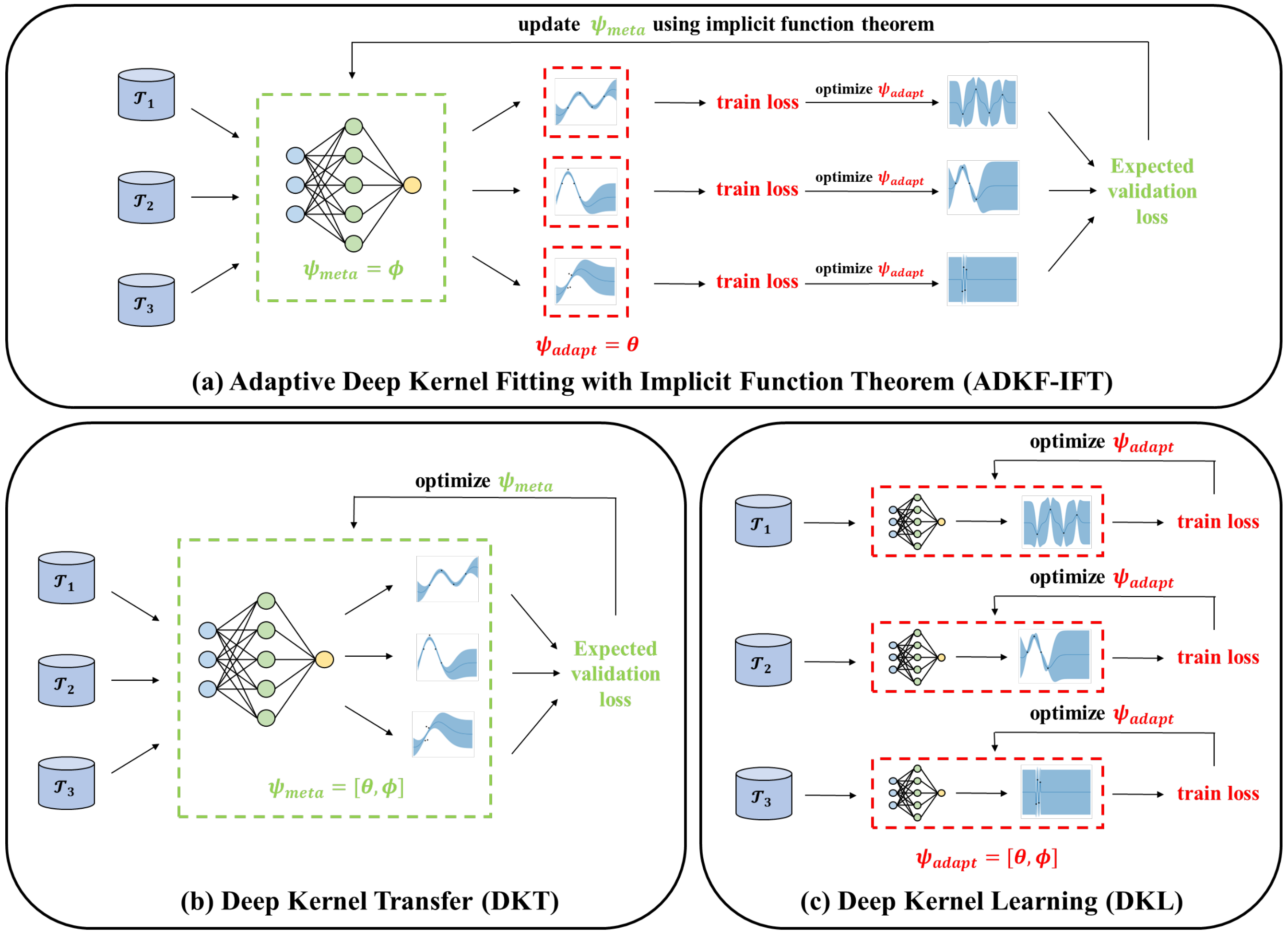
(c) Deep Kernel Learning (DKL)

Contrast DKL, DKT and ADKF-IFT


update ψ_{meta} using implicit function theorem
$\mathcal{T}_1$
$\mathcal{T}_2$
$\mathcal{T}_3$
$\psi_{meta} = \phi$
train loss
optimize ψ_{adapt}
Expected validation loss
$\psi_{adapt} = \theta$
(a) Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT)
optimize ψ_{meta}
$\psi_{meta} = [\theta, \phi]$
(b) Deep Kernel Transfer (DKT)
$\psi_{adapt} = [\theta, \phi]$
(c) Deep Kernel Learning (DKL)

Justification: two related tasks are more likely to have different noise levels, signal variances, or characteristic lengthscales than to require substantially different feature representations.

Contrast DKL, DKT and ADKF-IFT


update ψ_{meta} using implicit function theorem
$\mathcal{T}_1$
$\mathcal{T}_2$
$\mathcal{T}_3$
$\psi_{meta} = \phi$
train loss
optimize ψ_{adapt}
Expected validation loss
$\psi_{adapt} = \theta$
(a) Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT)
optimize ψ_{meta}
$\psi_{meta} = [\theta, \phi]$
Expected validation loss
(b) Deep Kernel Transfer (DKT)
optimize ψ_{adapt}
train loss
$\psi_{adapt} = [\theta, \phi]$
(c) Deep Kernel Learning (DKL)

- **ADKF-IFT** reduces overfitting:
  - ✓ It regularizes the feature extractor using meta-learning.
  - ✓ It learns feature extractor and base kernel parameters on different subsets (train/test) of a dataset.
- **ADKF-IFT** reduces underfitting:
  - ✓ It adapts base kernel parameters separately to each task.

Justification: two related tasks are more likely to have different noise levels, signal variances, or characteristic lengthscales than to require substantially different feature representations.

How to Solve the Bilevel Optimization Problem?

- **Meta-training**: ADKF-IFT can be formalized as a bi-level optimization problem.

$$\begin{aligned} \psi^*_{\text{meta}} &= \arg\min_{\boldsymbol{\psi}_{\text{meta}}} \ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}), \mathcal{T})], \\ \text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}) &= \arg\min_{\boldsymbol{\psi}_{\text{adapt}}} \ \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}). \longleftarrow \end{aligned}$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

How to Solve the Bilevel Optimization Problem?

- **Meta-training**: ADKF-IFT can be formalized as a bi-level optimization problem.

$$\psi^*_{\text{meta}} = \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}), \mathcal{T})],$$

$$\text{s.t.}\quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}}) = \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}). \longleftarrow$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Inner optimization:** the gradient of the train loss $\mathcal{L}_T$ can be calculated using auto-diff.

How to Solve the Bilevel Optimization Problem?

- **Meta-training**: ADKF-IFT can be formalized as a bi-level optimization problem.

implicit function of $\boldsymbol{\psi}_{meta}$ alone

$$\psi^*_{\text{meta}} = \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})],$$

$$\text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_\mathcal{T}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}).$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Inner optimization:** the gradient of the train loss $\mathcal{L}_\mathcal{T}$ can be calculated using auto-diff.

- **Outer optimization:** how to calculate the gradient for the validation loss $\mathcal{L}_V$?

Hypergradient:

$$\frac{d\,\mathcal{L}_V}{d\,\psi_{\text{meta}}} = \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi_{\text{meta}}}}_{\text{easy}} + \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi^*_{\text{adapt}}}}_{\text{easy}} \underbrace{\frac{\partial\,\psi^*_{\text{adapt}}}{\partial\,\psi_{\text{meta}}}}_{\text{hard}},$$

(by chain rule)

How to Solve the Bilevel Optimization Problem?

- **Meta-training**: ADKF-IFT can be formalized as a **bi-level optimization** problem.

implicit function of $\boldsymbol{\psi}_{meta}$ alone

$$\psi^*_{\text{meta}} = \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})],$$

$$\text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_\mathcal{T}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}). \longleftarrow$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Inner optimization:** the gradient of the train loss $\mathcal{L}_\mathcal{T}$ can be calculated using auto-diff.

- **Outer optimization:** how to calculate the gradient for the validation loss $\mathcal{L}_V$?

Hypergradient:

$$\frac{d\,\mathcal{L}_V}{d\,\psi_{\text{meta}}} = \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi_{\text{meta}}}}_{\textbf{easy}} + \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi^*_{\text{adapt}}}}_{\textbf{easy}}\underbrace{\frac{\partial\,\psi^*_{\text{adapt}}}{\partial\,\psi_{\text{meta}}}}_{\textbf{hard}},$$

(by chain rule)

✘ The best response function $\boldsymbol{\psi}^*_{adapt}$ is defined by an **argmin function!** How to differentiate it?

✘ Auto-diff requires tracking the gradients through many iterations of the inner optimization (**intractable**)!

Solve the Bilevel Optimization Problem by Implicit Function Theorem

- **Meta-training**: ADKF-IFT can be formalized as a **bi-level optimization** problem.

implicit function of $\boldsymbol{\psi}_{meta}$ alone

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

$$\text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \arg\min_{\psi_{\text{adapt}}} \mathcal{L}_\mathcal{T}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}). \longleftarrow$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Outer optimization:** how to calculate the gradient for the validation loss $\mathcal{L}_V$?

Hypergradient:

$$\frac{d\,\mathcal{L}_V}{d\,\psi_{\text{meta}}} = \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi_{\text{meta}}}}_{\textbf{easy}} + \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi^*_{\text{adapt}}}}_{\textbf{easy}} \underbrace{\boxed{\frac{\partial\,\psi^*_{\text{adapt}}}{\partial\,\psi_{\text{meta}}}}}_{\textbf{hard}}, \qquad \text{(by chain rule)}$$

- Since $\boldsymbol{\psi}^*_{adapt}$ is **a critical point of the train loss** $\mathcal{L}_\mathcal{T}$, we can apply the **Implicit Function Theorem** (**IFT**)!

Solve the Bilevel Optimization Problem by Implicit Function Theorem

- **Meta-training**: **ADKF-IFT** can be formalized as a **bi-level optimization** problem.

implicit function of $\boldsymbol{\psi}_{meta}$ alone

$$\psi^*_{\text{meta}} = \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})],$$

$$\text{s.t.}\quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_\mathcal{T}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}). \longleftarrow$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Outer optimization:** how to calculate the gradient for the validation loss $\mathcal{L}_V$?

Hypergradient:

$$\frac{d\,\mathcal{L}_V}{d\,\psi_{\text{meta}}} = \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi_{\text{meta}}}}_{\text{easy}} + \underbrace{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi^*_{\text{adapt}}}}_{\text{easy}} \underbrace{\boxed{\frac{\partial\,\psi^*_{\text{adapt}}}{\partial\,\psi_{\text{meta}}}}}_{\text{hard}},$$

(by chain rule)

- Since $\boldsymbol{\psi}^*_{adapt}$ is **a critical point of the train loss** $\mathcal{L}_\mathcal{T}$, we can apply the **Implicit Function Theorem** (**IFT**)!

IFT:

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

where $\psi'_{\text{adapt}} = \psi^*_{\text{adapt}}(\psi'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$.

Exact and Efficient Gradient Computation

- **Meta-training**: ADKF-IFT can be formalized as a bi-level optimization problem.

implicit function of $\boldsymbol{\psi}_{meta}$ alone

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

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

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

Exact and Efficient Gradient Computation

- **Meta-training**: ADKF-IFT can be formalized as a **bi-level optimization** problem.

implicit function of $\boldsymbol{\psi}_{meta}$ alone

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

$$\text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \arg\min_{\psi_{\text{adapt}}} \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}).$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Inner optimization:** the gradient of the train loss $\mathcal{L}_T$ can be calculated using auto-diff.
  - ✓ No backpropagate through the feature extractor is required!
  - ✓ Use L-BFGS for base kernel parameter $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ optimization (fast and efficient).

Exact and Efficient Gradient Computation

- **Meta-training**: ADKF-IFT can be formalized as a bi-level optimization problem.

implicit function of $\boldsymbol{\psi}_{meta}$ alone

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

$$\text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \arg\min_{\psi_{\text{adapt}}} \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}).$$

best response function for a given task $\mathcal{T}$ and $\boldsymbol{\psi}_{meta}$

- **Inner optimization:** the gradient of the train loss $\mathcal{L}_T$ can be calculated using auto-diff.

  ✓ No backpropagate through the feature extractor is required!

  ✓ Use L-BFGS for base kernel parameter $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$ optimization (fast and efficient).

- **Outer optimization:** the hypergradient of the validation loss $\mathcal{L}_V$ can be obtained using IFT.

IFT:

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

  ✓ Common GP based kernels (e.g., RBF) contains only a handful of parameters $\boldsymbol{\psi}_{adapt} = \boldsymbol{\theta}$.

  ✓ The inverse Hessian in IFT can be computed exactly without any approximation!

General Framework vs. Specific Instantiations

ADKF-IFT can be formalized as a **bi-level optimization** problem:

$$
\begin{aligned}
\psi^*_{\text{meta}} &= \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})], \\
\text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) &= \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}).
\end{aligned}
$$

- ✓ $\boldsymbol{\psi}_{adapt}, \boldsymbol{\psi}_{meta}, \mathcal{L}_T, \mathcal{L}_V$ could be anything, which makes ADKF-IFT a **general framework.**

- ✓ Any particular choice of $\boldsymbol{\psi}_{adapt}, \boldsymbol{\psi}_{meta}, \mathcal{L}_T, \mathcal{L}_V$ is an **instantiation** of the general framework.

- ✓ DKL and DKT are special instantiations **(extreme cases)** of this general framework!

The General Framework Unifies Previous Methods (DKL and DKT)

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

$$\begin{aligned} \psi^*_{\text{meta}} &= \underset{\psi_{\text{meta}}}{\arg\min}\ \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})], \\ \text{s.t.} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) &= \underset{\psi_{\text{adapt}}}{\arg\min}\ \mathcal{L}_\mathcal{T}(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}). \end{aligned}$$

$\boldsymbol{\psi}_{adapt} = \boldsymbol{\psi} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$

$\boldsymbol{\psi}_{meta} = \emptyset$

$\mathcal{L}_T$ = NLML

$\mathcal{L}_V$ = NLML

$\boldsymbol{\psi}_{meta} = \boldsymbol{\psi} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$

$\boldsymbol{\psi}_{adapt} = \emptyset$

$$\psi^* = \underset{\psi}{\arg\min}\, \text{NLML}\,(\psi, \mathcal{S}_\mathcal{T})$$

Deep Kernel Learning (DKL)

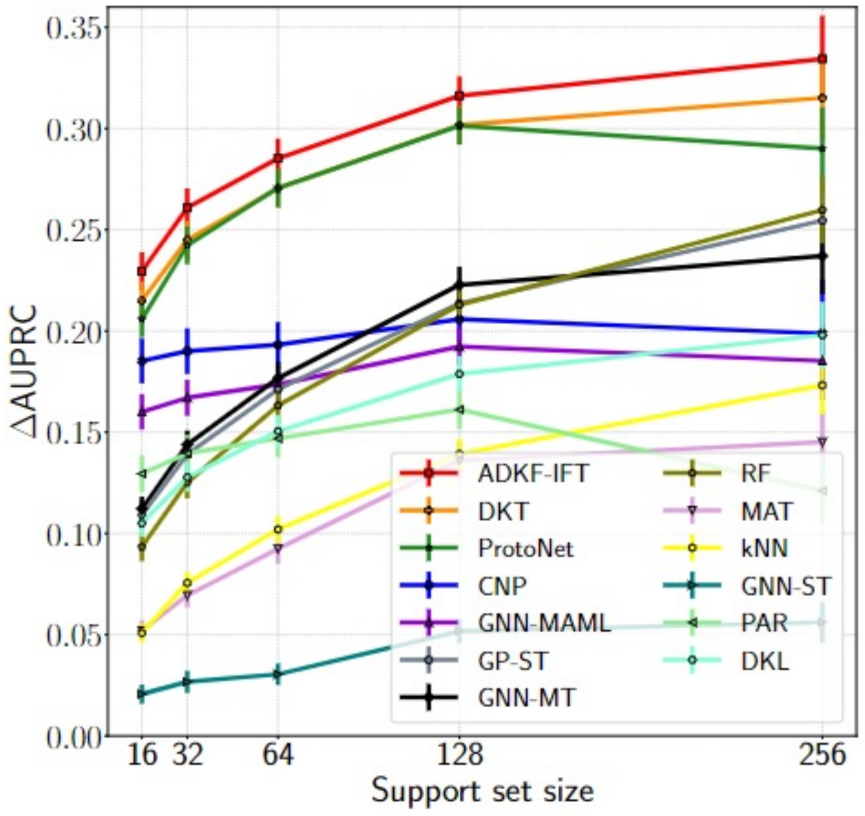
$$\psi^* = \underset{\psi}{\arg\min}\, \mathbb{E}_{p(\mathcal{T})}[\text{NLML}(\psi, \mathcal{T})]$$

Deep Kernel Transfer (DKT)

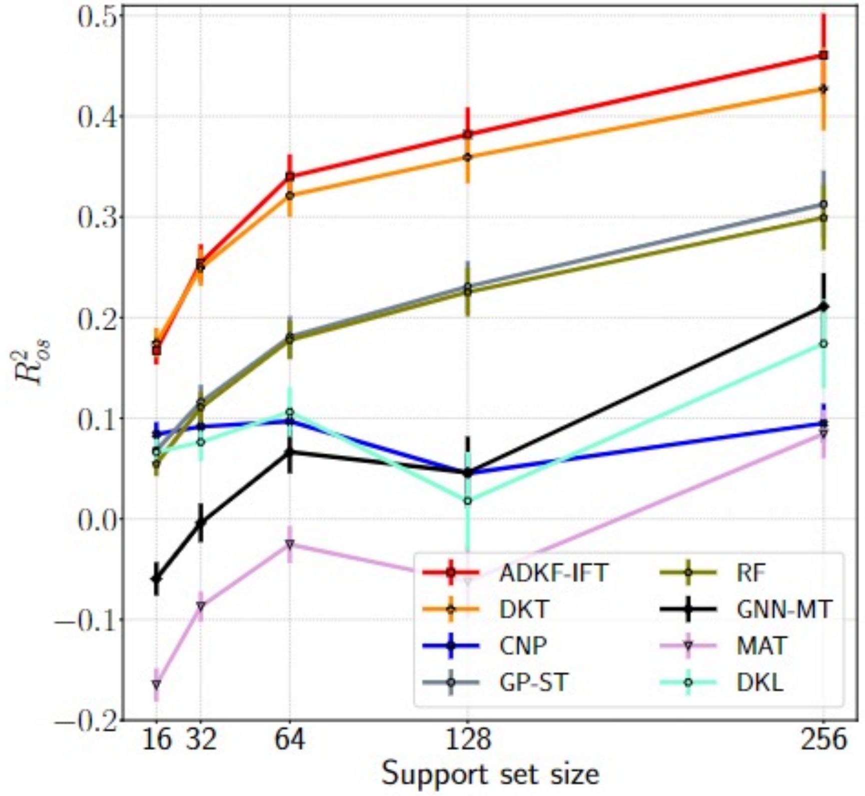
(**NLML**: negative log marginal likelihood)

Experiment 1: Few-shot Molecular Property Prediction on FS-Mol

- **FS-Mol** (Stanley et al., 2021): 4,938 training tasks, 40 validation tasks, 157 test tasks; 233,786 unique compounds.



(a) Classification (157 tasks).

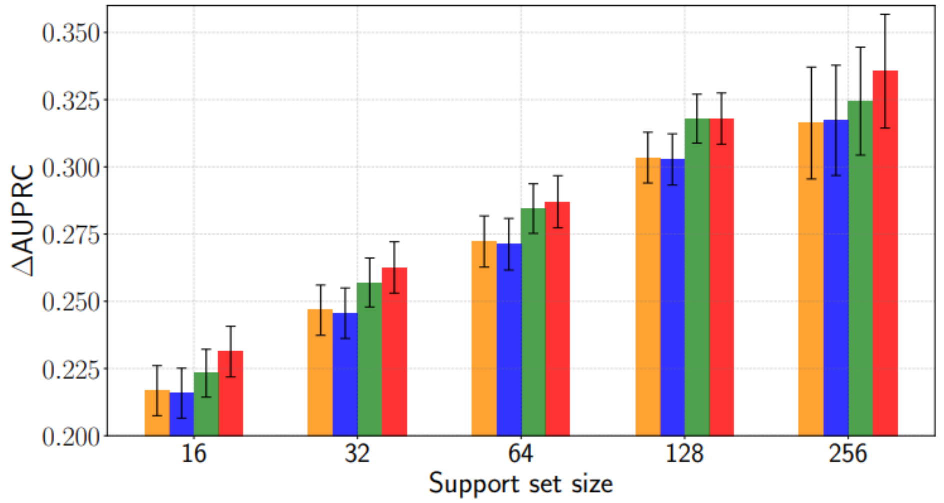


(b) Regression (111 tasks).

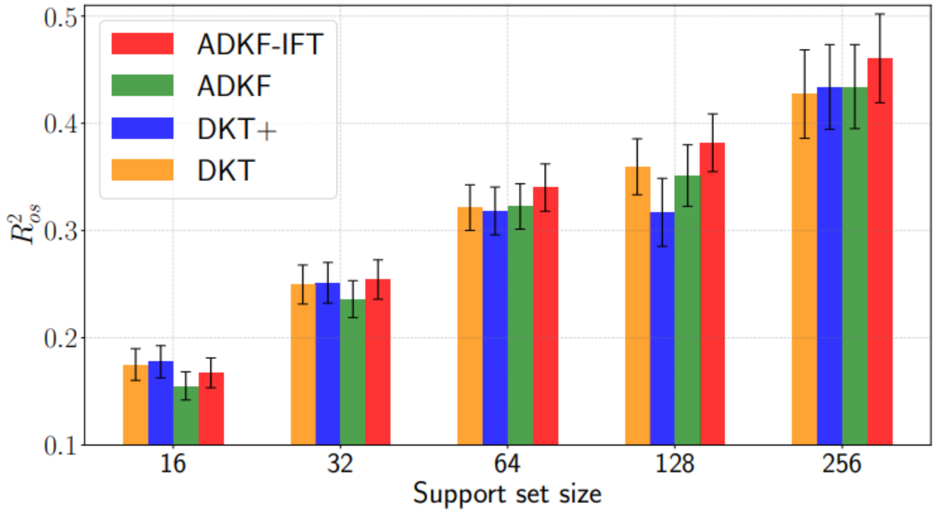
✓ The improvements of ADKF-IFT over other methods are **statistically significant!**

Stanley, Megan, et al. "Fs-mol: A few-shot learning dataset of molecules." Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2). 2021.

Experiment 1: Ablation Study and Analysis on FS-Mol



(a) Classification (157 tasks).



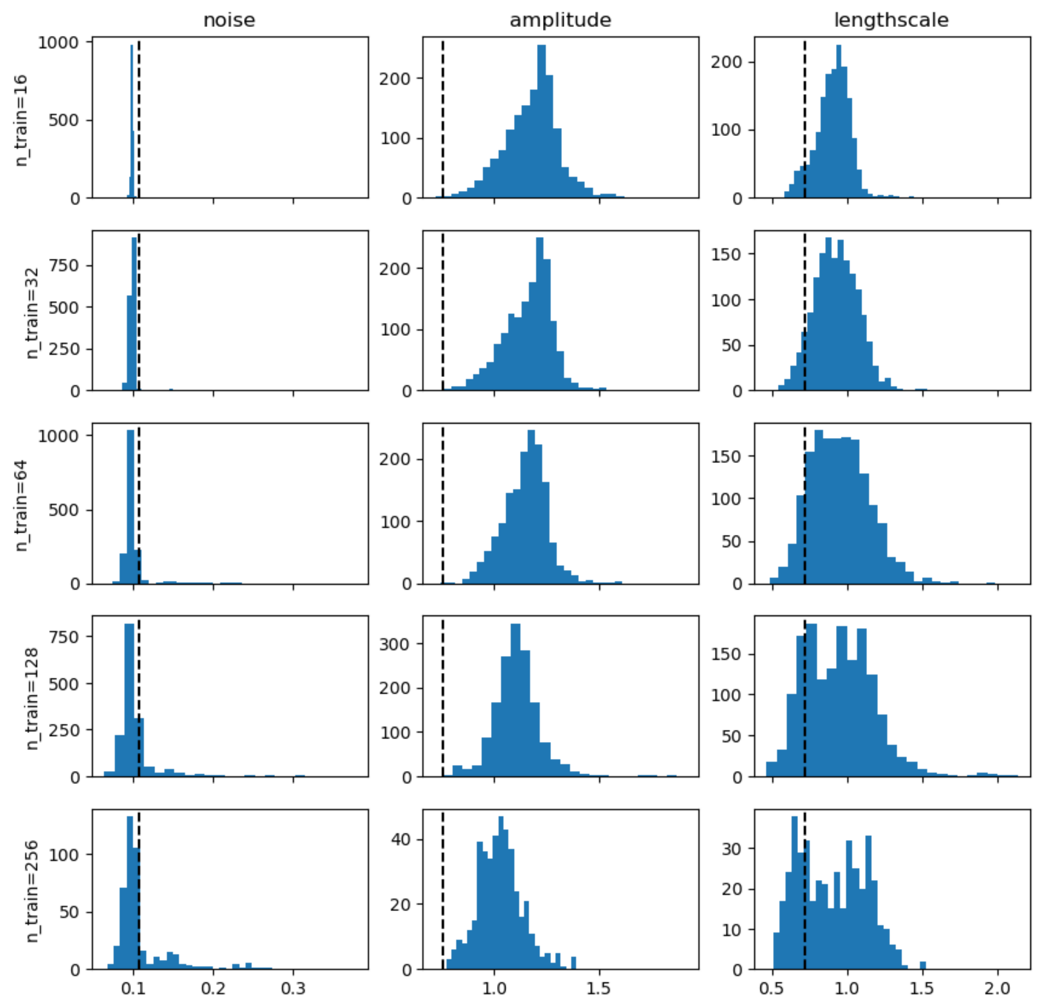
(b) Regression (111 tasks).

✓ **DKT ≈ DKT+ ≤ ADKF < ADKF+**

- **DKT+** is like **DKT** but tuning the base kernel parameters $\boldsymbol{\theta}$ for each task during meta-testing.
- **ADKF** is like **ADKF+** but ignoring the gradient through the best response function $\boldsymbol{\psi}^*_{\boldsymbol{adapt}}$.

$$\frac{d\,\mathcal{L}_V}{d\,\psi_{\text{meta}}} = \frac{\partial\,\mathcal{L}_V}{\partial\,\psi_{\text{meta}}} + \cancelto{\mathbf{0}}{\frac{\partial\,\mathcal{L}_V}{\partial\,\psi^*_{\text{adapt}}}\frac{\partial\,\psi^*_{\text{adapt}}}{\partial\,\psi_{\text{meta}}}},$$

Experiment 1: Ablation Study and Analysis on FS-Mol



- Blue histogram: the distribution of the base kernel parameters $\boldsymbol{\theta}$ across different tasks learned by ADKF-IFT.

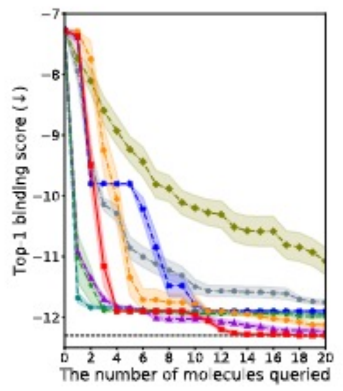
- Dotted vertical line: the base kernel parameters $\boldsymbol{\theta}$ shared across all tasks learned by DKT.

✓ The base kernel parameters $\boldsymbol{\theta}$ **do vary across tasks!**

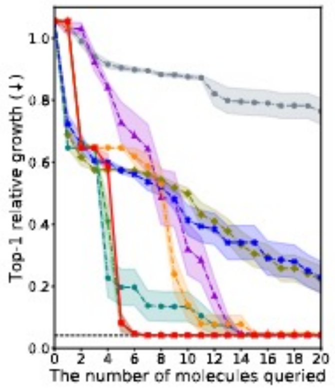
✓ ADKF-IFT achieves **better signal-to-noise ratio!**

Experiment 2: OOD Molecular Property Prediction and Optimization

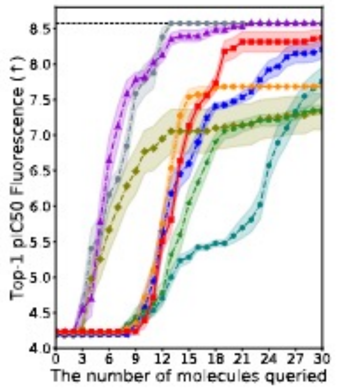
- Bayesian optimization (BO)



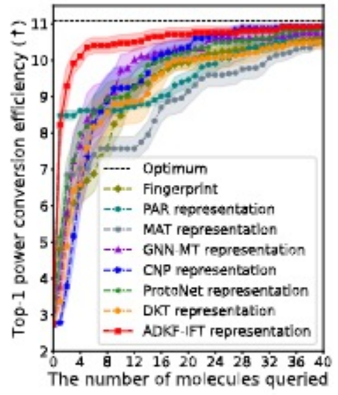
(a) Molecular docking.



(b) Antibiotic discovery.



(c) Antiviral drug design.



(d) Material design.

- **Surrogate model:** GP operating on top of the features extracted by DNNs meta-trained on FS-Mol by different methods.

- **Evaluation:** four OOD molecular design tasks outside of FS-Mol.

- Test predictive negative log likelihood (NLL)

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	**4.601 ± 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	**1.137 ± 0.011**	**1.496 ± 0.043**	4.781 ± 0.087	**0.996 ± 0.007**

✓ ADKF-IFT enables **fastest discovery of top performing molecules!**

✓ ADKF-IFT achieves **competitive test predictive performance!**

Summary

Our proposed **Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT)** approach:

✓ meta-learns feature representations that facilitate the adaptation of task-specific GP models;

✓ generalizes DKL and DKT for training deep kernel GPs using a bilevel optimization framework;

✓ efficiently solve the bilevel optimization problem by implicit function theorem;

✓ produces state-of-the-art results on few-shot molecular property prediction benchmarks;

✓ achieves great performance on OOD molecular property prediction and optimization tasks;

✓ produces well-calibrated models for fully-automated high-throughput experimentation that could accelerate drug discovery and material design.

Published at ICLR 23: https://openreview.net/forum?id=KXRSh0sdVTP

Thank you!

Published at ICLR 23: https://openreview.net/forum?id=KXRSh0sdVTP

Limitations and Future Work Directions

1. Use ARD for the lengthscale parameter in the base kernel for automatic feature selection.

2. Adapt the feature extractor to each task by allowing small deviations from a meta-learned prior.

3. Adopt a more principled approximate inference method for GP classification.

4. Inject domain expertise from drug discovery into the base kernel with hand-curated features and kernel combination.

5. Consider other application domains such as few-shot image classification.

Appendix 1: Mean ranks of Compared Methods on FS-Mol

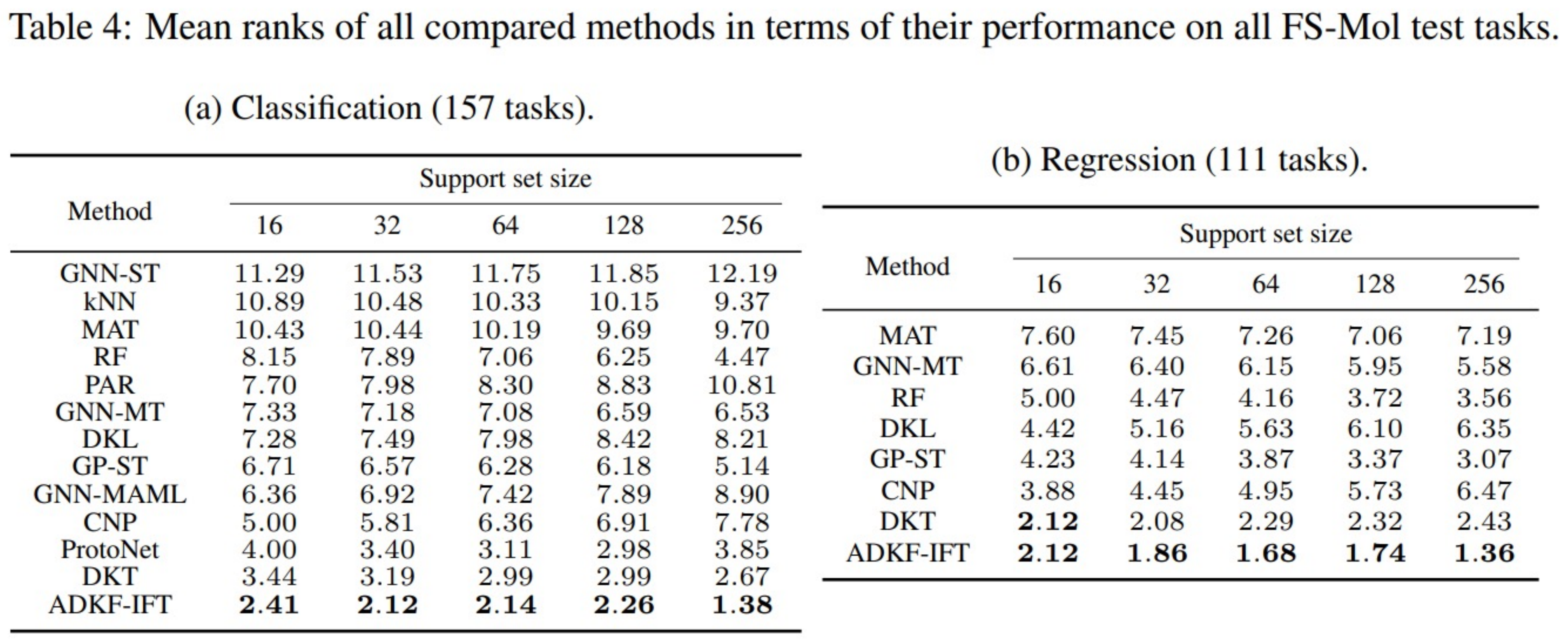


Table 4: Mean ranks of all compared methods in terms of their performance on all FS-Mol test tasks.

(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**

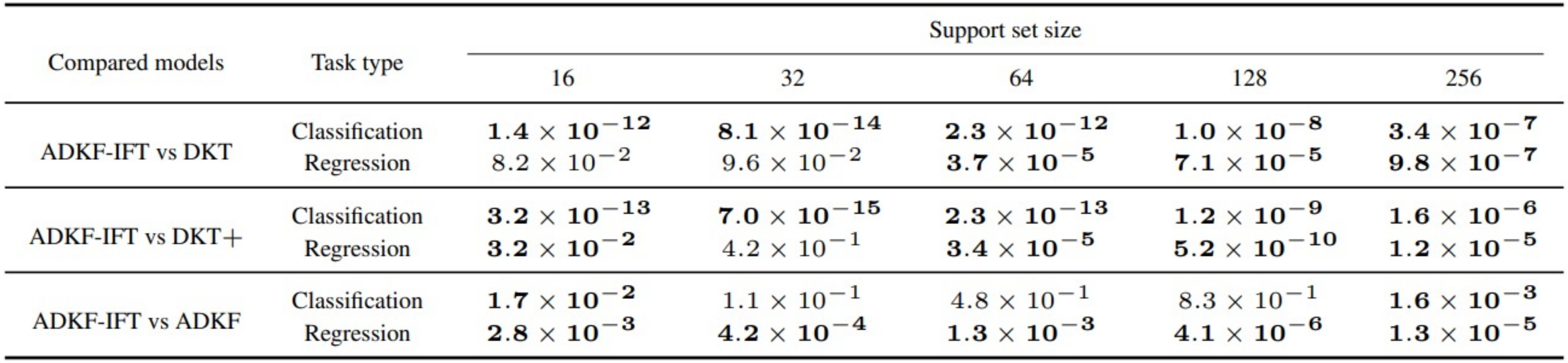
(b) Regression (111 tasks).

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**

ADKF-IFT **consistently ranks the best** in all settings!

Appendix 2: Statistical Comparisons on FS-Mol

Table 5: p-values from the two-sided Wilcoxon signed-rank test for statistical comparisons between ADKF-IFT and DKT/DKT+/ADKF. 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 $\alpha = 0.05$.



Compared models	Task type	Support set size				
		16	32	64	128	256
ADKF-IFT vs DKT	Classification	$\mathbf{1.4 \times 10^{-12}}$	$\mathbf{8.1 \times 10^{-14}}$	$\mathbf{2.3 \times 10^{-12}}$	$\mathbf{1.0 \times 10^{-8}}$	$\mathbf{3.4 \times 10^{-7}}$
	Regression	8.2×10^{-2}	9.6×10^{-2}	$\mathbf{3.7 \times 10^{-5}}$	$\mathbf{7.1 \times 10^{-5}}$	$\mathbf{9.8 \times 10^{-7}}$
ADKF-IFT vs DKT+	Classification	$\mathbf{3.2 \times 10^{-13}}$	$\mathbf{7.0 \times 10^{-15}}$	$\mathbf{2.3 \times 10^{-13}}$	$\mathbf{1.2 \times 10^{-9}}$	$\mathbf{1.6 \times 10^{-6}}$
	Regression	$\mathbf{3.2 \times 10^{-2}}$	4.2×10^{-1}	$\mathbf{3.4 \times 10^{-5}}$	$\mathbf{5.2 \times 10^{-10}}$	$\mathbf{1.2 \times 10^{-5}}$
ADKF-IFT vs ADKF	Classification	$\mathbf{1.7 \times 10^{-2}}$	1.1×10^{-1}	4.8×10^{-1}	8.3×10^{-1}	$\mathbf{1.6 \times 10^{-3}}$
	Regression	$\mathbf{2.8 \times 10^{-3}}$	$\mathbf{4.2 \times 10^{-4}}$	$\mathbf{1.3 \times 10^{-3}}$	$\mathbf{4.1 \times 10^{-6}}$	$\mathbf{1.3 \times 10^{-5}}$

The improvements of ADKF-IFT over other methods are **statistically significant!**

Appendix 3: Sub-benchmark Performance on FS-Mol

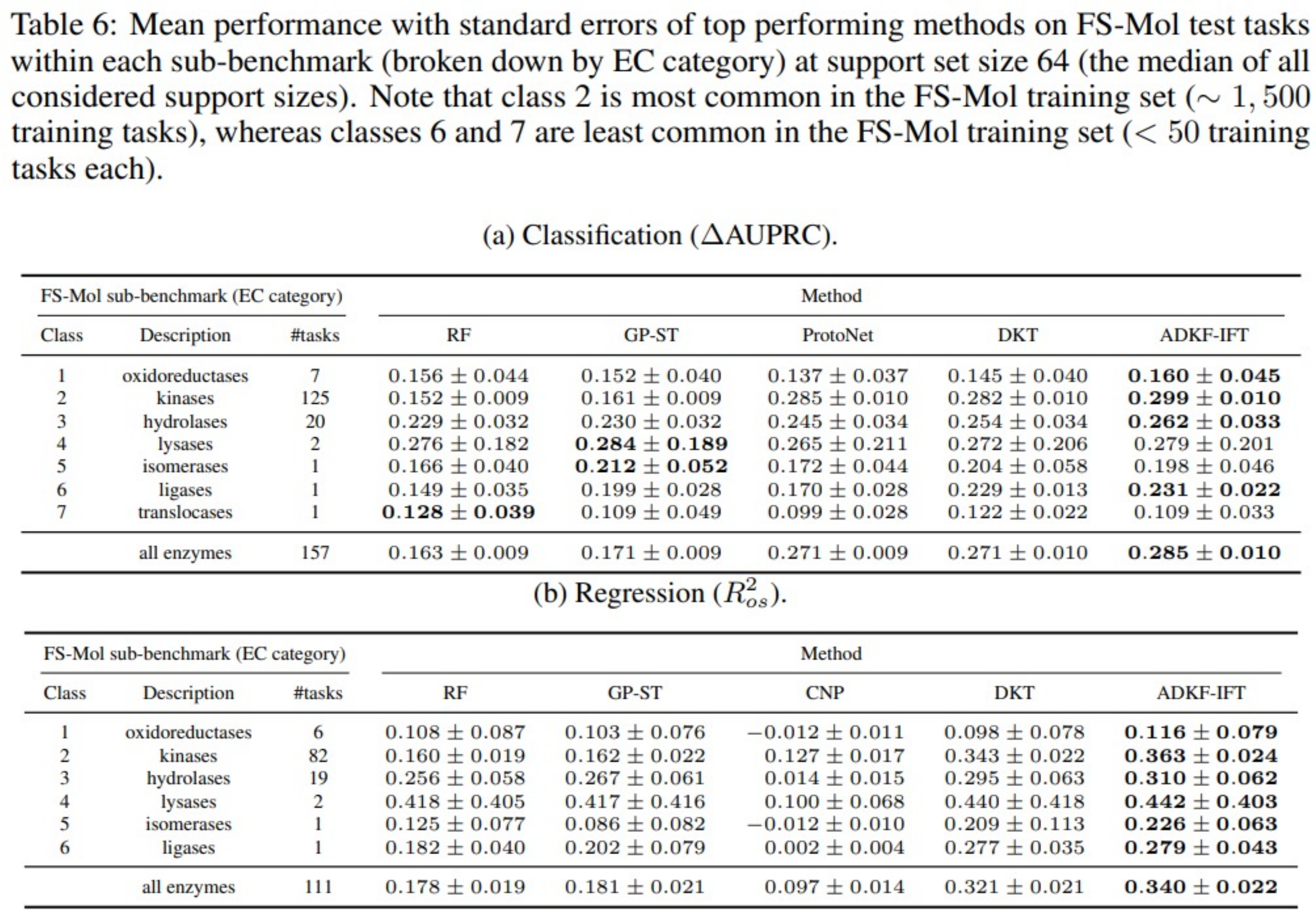


Table 6: Mean performance with standard errors of top performing methods on FS-Mol test tasks within each sub-benchmark (broken down by EC category) at support set size 64 (the median of all considered support sizes). Note that class 2 is most common in the FS-Mol training set ($\sim 1,500$ training 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	$\mathbf{0.340 \pm 0.022}$

Appendix 4: Meta-testing Cost on FS-Mol

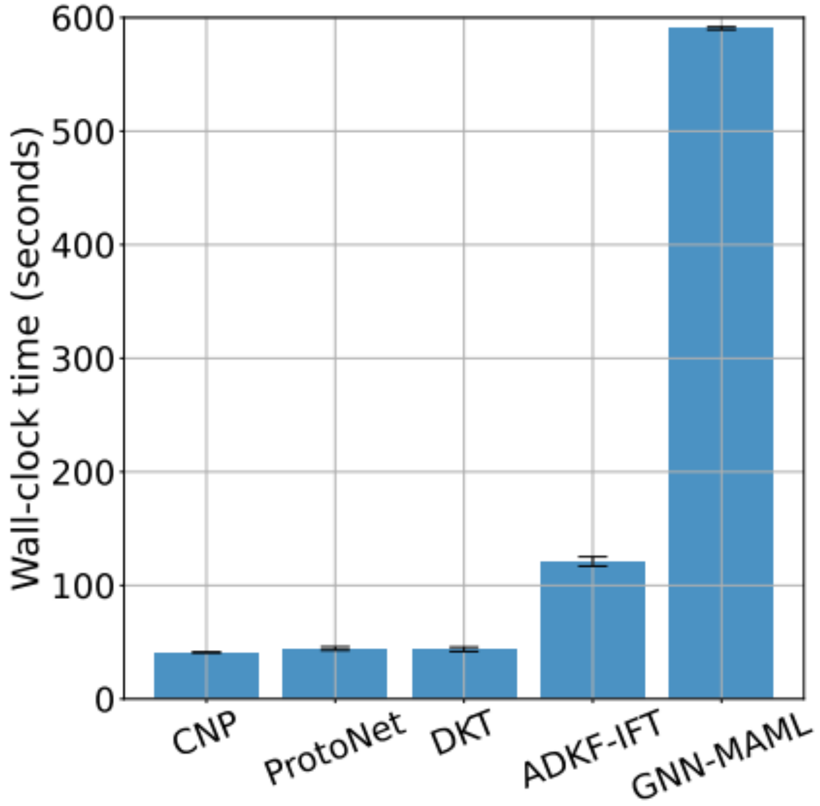


Figure 5: Wall-clock time consumed (with standard errors) when meta-testing on a pre-defined set of FS-Mol classification tasks using each of the compared meta-learning methods.

Appendix 6: Few-shot Molecular Property Prediction on MoleculeNet (Wu et al., 2018)

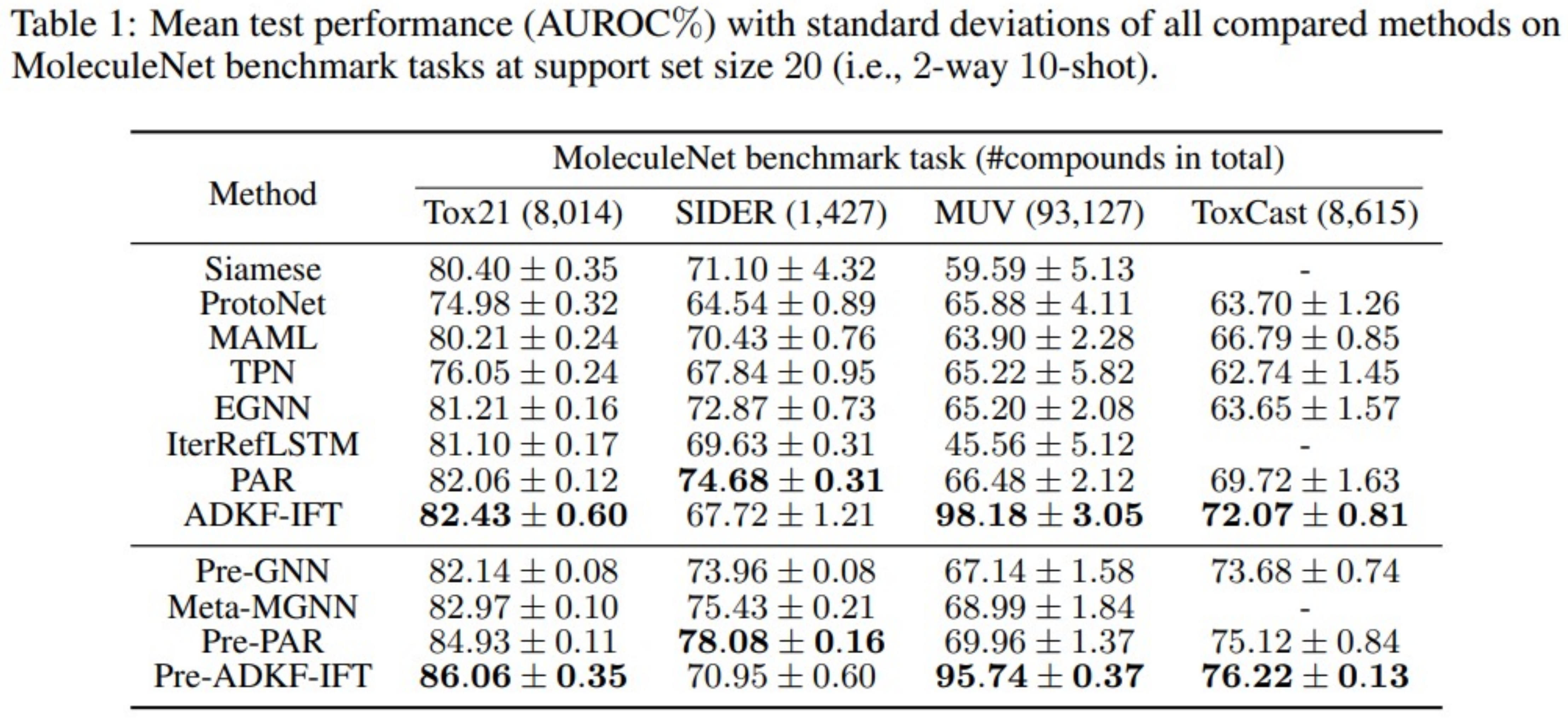


Table 1: Mean test performance (AUROC%) with standard deviations of all compared methods on MoleculeNet benchmark tasks at support set size 20 (i.e., 2-way 10-shot).

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	$\mathbf{95.74 \pm 0.37}$	$\mathbf{76.22 \pm 0.13}$

Wu, Zhenqin, et al. "MoleculeNet: a benchmark for molecular machine learning." Chemical science 9.2 (2018): 513-530.