Pin-Tuning: Parameter-Efficient In-Context Tuning for Few-Shot Molecular Property Prediction

Liang Wang[1 2] **Qiang Liu**[1 2 †] **Shaozhen Liu**[1] **Xin Sun**[3] **Shu Wu**[1 2] **Liang Wang**[1 2 3]

[1]New Laboratory of Pattern Recognition (NLPR)
State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS)
Institute of Automation, Chinese Academy of Sciences (CASIA)
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] University of Science and Technology of China

liang.wang@cripac.ia.ac.cn, qiang.liu@nlpr.ia.ac.cn, liushaozhen2025@ia.ac.cn
sunxin000@mail.ustc.edu.cn, {shu.wu, wangliang}@nlpr.ia.ac.cn

Abstract

Molecular property prediction (MPP) is integral to drug discovery and material science, but often faces the challenge of data scarcity in real-world scenarios. Addressing this, few-shot molecular property prediction (FSMPP) has been developed. Unlike other few-shot tasks, FSMPP typically employs a pre-trained molecular encoder and a context-aware classifier, benefiting from molecular pre-training and molecular context information. Despite these advancements, existing methods struggle with the ineffective fine-tuning of pre-trained encoders. We attribute this issue to the imbalance between the abundance of tunable parameters and the scarcity of labeled molecules, and the lack of contextual perceptiveness in the encoders. To overcome this hurdle, we propose a parameter-efficient in-context tuning method, named `Pin-Tuning`. Specifically, we propose a lightweight adapter for pre-trained message passing layers (`MP-Adapter`) and Bayesian weight consolidation for pre-trained atom/bond embedding layers (`Emb-BWC`), to achieve parameter-efficient tuning while preventing over-fitting and catastrophic forgetting. Additionally, we enhance the `MP-Adapters` with contextual perceptiveness. This innovation allows for in-context tuning of the pre-trained encoder, thereby improving its adaptability for specific FSMPP tasks. When evaluated on public datasets, our method demonstrates superior tuning with fewer trainable parameters, improving few-shot predictive performance.[‡]

1 Introduction

In the field of drug discovery and material science, molecular property prediction (MPP) stands as a pivotal task [5, 9, 63]. MPP involves the prediction of molecular properties like solubility and toxicity, based on their structural and physicochemical characteristics, which is integral to the development of new pharmaceuticals and materials. However, a major challenge encountered in real-world MPP scenarios is data scarcity. Obtaining extensive molecular data with well-characterized properties can be time-consuming and expensive. To address this, few-shot molecular property prediction (FSMPP) has emerged as a crucial approach, enabling predictions with limited labeled molecules [1, 41, 4].

The methodology for general MPP typically adheres to an encoder-classifier framework [71, 23, 27, 56], as illustrated in Figure 2(a). In this streamlined framework, the encoder converts molecular

---

[†]Corresponding author
[‡]Code is available at: `https://github.com/CRIPAC-DIG/Pin-Tuning`

structures into vectorized representations [12, 28, 50, 67, 2], and then the classifier uses these representations to predict molecular properties. In the context of few-shot scenarios, two significant discoveries have been instrumental in advancing this task. Firstly, *pre-trained molecular encoders* have demonstrated consistent effectiveness in FSMPP tasks [20, 14, 58]. This indicates the utility of leveraging pre-acquired knowledge in dealing with data-limited scenarios. Secondly, unlike typical few-shot tasks such as image classification [57, 48], FSMPP tasks greatly benefits from *molecular context information*. This involves comprehending the seen many-to-many relationships between molecules and properties [58, 45, 73], as molecules are multi-labeled by various properties. These two discoveries have collectively led to the development of the widely used FSMPP framework that utilizes a pre-trained encoder followed by a context-aware classifier, as shown in Figure 2(b).

Despite the progress, there are observed limitations in the current approaches to FSMPP. Notably, while using a pre-trained molecular encoder generally outperforms training from scratch, fine-tuning the pre-trained encoder often leads to inferior results compared to keeping it frozen, which can be observed in Figure 1.

The observed ineffective fine-tuning can be attributed to two primary factors: (i) Imbalance between the abundance of tunable parameters and the scarcity of labeled molecules: fine-tuning all parameters of a pre-trained encoder with few labeled molecules leads to a disproportionate ratio of tunable parameters to available data. This imbalance often results in over-fitting and catastrophic forgetting [7, 6]. (ii) Limited contextual perceptiveness in the encoder: while molecular context is leveraged to enhance the classifier [58, 73], the encoder typically lacks the explicit capability to perceive this context, relying instead on implicit gradient-based optimization. This leads to the encoder not directly engaging with the nuanced molecular context information that is critical in FSMPP tasks. In summary, while significant strides have been made, the challenges of imbalance between the number of parameters and labeled data, along with the need for contextual perceptiveness in the encoder, necessitate more sophisticated methodologies in this domain.

Figure 1: Comparison of molecular encoders trained via different paradigms: train-from-scratch, pretrain-then-freeze, and pretrain-then-finetune. The evaluation is conducted across two datasets and three encoder architectures [20, 47, 66]. The results consistently demonstrate that while pretraining outperforms training from scratch, the current methods do not yet effectively facilitate finetuning.

Based on the aforementioned analysis, we propose the parameter-efficient in-context tuning method, named `Pin-Tuning`, to address the two primary challenges in FSMPP. To overcome the parameter-data imbalance, we propose a parameter-efficient chemical knowledge adaptation approach for pre-trained molecular encoders. A lightweight adapters (`MP-Adapter`) are designed to tune the pre-trained message passing layers efficiently. Additionally, we impose a Bayesian weight consolidation (`Emb-BWC`) on the pre-trained embedding layers to prevent aggressive parameter updates, thereby mitigating the risk of over-fitting and catastrophic forgetting. To address the second challenge, we further endow the `MP-Adapter` with the capability to perceive context. This innovation allows for in-context tuning of the pre-trained molecular encoders, enabling them to adapt more effectively to specific downstream tasks. Our approach is rigorously evaluated on public datasets. The experimental results demonstrate that our method achieves superior tuning performance with fewer trainable parameters, leading to enhanced performance in few-shot molecular property prediction.

The main contributions of our work are summarized as follows:

- We analyze the deficiencies of existing FSMPP approaches regarding the adaptation of pre-trained molecular encoders. The key issues include an imbalance between the number of tunable parameters and labeled molecules, as well as a lack of contextual perceptiveness in the encoders.

- We propose `Pin-Tuning` to adapt the pre-trained molecular encoders for FSMPP tasks. This includes the `MP-Adapter` for message passing layers and the `Emb-BWC` for embedding layers, facilitating parameter-efficient tuning of pre-trained molecular encoders.

- We further endow the `MP-Adapter` with the capability to perceive context to allows for in-context tuning, which provides more meaningful adaptation guidance during the tuning process.

- We conduct extensive experiments on benchmark datasets, which show that `Pin-Tuning` outperforms state-of-the-art methods on FSMPP by effectively tuning pre-trained molecular encoders.

2

2 Related work

Few-shot molecular property prediction. Few-shot molecular property prediction aims to accurately predict the properties of new molecules with limited training data [49]. Early research applied general few-shot techniques to FSMPP. IterRefLSTM [1] is the pioneer work to leverage metric learning to solve FSMPP problem. Following this, Meta-GGNN [41] and Meta-MGNN [14] introduce meta-learning with graph neural networks, setting a foundational framework that subsequent studies have continued to build upon [39, 40, 4]. It is noteworthy that Meta-MGNN employs a *pre-trained molecular encoder* [20] and achieves superior results through fine-tuning in the meta-learning process compared to training from scratch. In fact, pre-trained graph neural networks [64, 36, 17, 54, 37] have shown promise in enhancing various graph-based downstream tasks [52, 13], including molecular property prediction [60, 62, 38, 72]. Recent efforts have shifted towards leveraging unique nature in FSMPP, such as the many-to-many relationships between molecules and properties arising from the multi-labeled nature of molecules, often referred to as the *molecular context*. PAR [58] initially employs graph structure learning [32, 55] to connect similar molecules through a homogeneous context graph. MHNfs [45] introduces a large-scale external molecular library as context to augment the limited known information. GS-Meta [73] further incorporates auxiliary task to depict the many-to-many relationships.

Parameter-efficient tuning. As pre-training techniques have advanced, tuning of pre-trained models has become increasingly crucial. Traditional full fine-tuning approaches updates all parameters, often leading to high computational costs and the risk of over-fitting, especially when available data for downstream tasks are limited [33, 15]. This challenge has led to the emergence of parameter-efficient tuning [26, 29, 34]. The philosophy of parameter-efficient tuning is to optimize a small subset of parameters, reducing the computational costs while retaining or even improving performance on downstream tasks [19, 69]. Among the various strategies, the adapters [18, 42, 59] have gained prominence. Adapters are small modules inserted between the pre-trained layers. During the tuning process, only the parameters of these adapters are updated while the rest remains frozen, which not only improves tuning efficiency but also offers an elegant solution to the generalization [70, 30, 8]. By keeping the majority of the pre-trained parameters intact, adapters preserve the rich pre-trained knowledge. This attribute is particularly valuable in many real-world applications including FSMPP.

3 Preliminaries

3.1 Problem formulation

Let $\{\mathcal{T}\}$ be a collection of tasks, where each task $\mathcal{T}$ involves the prediction of a property p. The training set comprising multiple tasks $\{\mathcal{T}_{\text{train}}\}$, is represented as $\mathcal{D}_{\text{train}} = \{(m_i, y_{i,t}) | t \in \{\mathcal{T}_{\text{train}}\}\}$, with m_i indicating a molecule and $y_{i,t}$ its associated label for task t. Correspondingly, the test set $\mathcal{D}_{\text{test}}$, formed by tasks $\{\mathcal{T}_{\text{test}}\}$, ensures a separation of properties between training and testing phases, as the property sets $\{p_{\text{train}}\}$ and $\{p_{\text{test}}\}$ are disjoint ($\{p_{\text{train}}\} \cap \{p_{\text{test}}\} = \emptyset$).

The goal of FSMPP is to train a model using $\mathcal{D}_{\text{train}}$ that can accurately infer new properties from a limited number of labeled molecules in $\mathcal{D}_{\text{test}}$. Episodic training has emerged as a promising strategy in meta-learning [10, 16] to deal with few-shot problem. Instead of retaining all $\{\mathcal{T}_{\text{train}}\}$ tasks in memory, episodes $\{E_t\}_{t=1}^{B}$ are iteratively sampled throughout the training process. For each episode E_t, a particular task $\mathcal{T}_t$ is selected from the training set, along with corresponding support set $\mathcal{S}_t$ and query set $\mathcal{Q}_t$. Typically, the prediction task involves classifying molecules into two classes: *positive* ($y = 1$) or *negative* ($y = 0$). Then a 2-way K-shot episode $E_t = (\mathcal{S}_t, \mathcal{Q}_t)$ is constructed. The support set $\mathcal{S}_t = \{(m_i^s, y_{i,t}^s)\}_{i=1}^{2K}$ includes $2K$ examples, each class contributing K molecules. The query set containing M molecules is denoted as $\mathcal{Q}_t = \{(m_i^q, y_{i,t}^q)\}_{i=1}^{M}$.

3.2 Encoder-classifier framework for FSMPP

Encoder-classifier framework is widely adopted in FSMPP methods. As illustrated in Figure 2(a), given a molecule m whose property need to be predicted, a molecular encoder $f(\cdot)$ first learns the molecule's representation based on its structure, i.e., $\boldsymbol{h}_m = f(m) \in \mathbb{R}^d$. The molecule m is generally represented as a graph $m = (\mathcal{V}, \mathbf{A}, \mathbf{X}, \mathbf{E})$, where $\mathcal{V}$ denotes the nodes (atoms), $\mathbf{A}$ represents the adjacent matrix defined by edges (chemical bonds), and $\mathbf{X}, \mathbf{E}$ denote the original feature of atoms

and bonds, then graph neural networks (GNNs) are employed as the molecular encoders [44, 51, 21]. Subsequently, the learned molecular representation is fed into a classifier $g(\cdot)$ to obtain the prediction $\hat{y} = g(\boldsymbol{h}_m)$. The model is trained by minimizing the discrepancy between $\hat{y}$ and the ground truth y.

Further, two key discoveries have been pivotal for FSMPP. The first is the proven effectiveness of pre-trained molecular encoders, while the second is the significant advantage gained from molecular context. Together, these discoveries have further reshaped the widely adopted FSMPP framework, which combines a *pre-trained encoder* followed by a *context-aware classifier*, as shown in Figure 2(b).

3.3 Pre-trained molecular encoders (PMEs)

Due to the scarcity of labeled data in molecular tasks, molecular pre-training has emerged as a crucial area, which involves training encoders on extensive molecular datasets to extract informative representations. Pre-GNN [20] is a classic pre-trained molecular encoder that has been widely used in addressing FSMPP tasks [14, 58, 73]. The backbone of Pre-GNN is a modified version of Graph Isomorphism Network (GIN) [65] tailored to molecules, which we call GIN-Mol, consisting of multiple atom/bond embedding layers and message passing layers.

Atom/Bond embedding layers. The raw atom features and bond features are both categorical vectors, denoted as $(i_{v,1}, i_{v,2}, \ldots, i_{v,|E_n|})$ and $(j_{e,1}, j_{e,2}, \ldots, j_{e,|E_e|})$ for atom v and bond e, respectively. These categorical features are embedded as:

$$\boldsymbol{h}_v^{(0)} = \sum_{a=1}^{|E_n|} \text{EmbAtom}_a(i_{v,a}), \quad \boldsymbol{h}_e^{(l)} = \sum_{b=1}^{|E_e|} \text{EmbBond}_b^{(l)}(j_{e,b}), \tag{1}$$

where $\text{EmbAtom}_a(\cdot)_{a \in \{1,\ldots,|E_n|\}}$ and $\text{EmbBond}_b(\cdot)_{b \in \{1,\ldots,|E_e|\}}$ represent embedding operations that map integer indices to d-dimensional real vectors, i.e., $\boldsymbol{h}_v^{(0)}, \boldsymbol{h}_e^{(l)} \in \mathbb{R}^d$, $l \in \{0, 1, \ldots, L-1\}$ represents the index of encoder layers, and L is the number of encoder layers. The atom embedding layer is present only in the first encoder layer, while an bond embedding layer exists in each layer.

Message passing layers. At the l-th encoder layer, atom representations are updated by aggregating the features of neighboring atoms and chemical bonds:

$$\boldsymbol{h}_v^{(l)} = \text{ReLU}\left(\text{MLP}^{(l)}\left(\sum_u \boldsymbol{h}_u^{(l-1)} + \sum_{e=(v,u)} \boldsymbol{h}_e^{(l-1)}\right)\right), \tag{2}$$

where $u \in \mathcal{N}(v) \cup \{v\}$ is the set of atoms connected to v, and $\boldsymbol{h}_v^{(l)} \in \mathbb{R}^d$ is the learned representation of atom v at the l-th layer. $\text{MLP}(\cdot)$ is implemented by 2-layer neural networks, in which the hidden dimension is d_1. After MLP, batch normalization is applied right before the ReLU. The molecule-level representation $\boldsymbol{h}_m \in \mathbb{R}^d$ is obtained by averaging the atom representations at the final layer.

4 The proposed Pin-Tuning method

This section delves into our motivation and proposed method. Our framework for FSMPP is depicted in Figure 2(c). The details of our principal design, `Pin-Tuning` for PMEs, is present in Figure 2(d).

As shown in Figure 1, pretraining then finetuning molecular encoders is a common approach. However, fully fine-tuning yields results inferior to simply freezing them. Thus, the following question arises:

> *How to effectively adapt pre-trained molecular encoders to downstream tasks, especially in few-shot scenarios?*

We analyze the reasons of observed ineffective fine-tuning issue, and attribute it to two primary factors: (i) imbalance between the abundance of tunable parameters and the scarcity of labeled molecules, and (ii) limited contextual perceptiveness in the encoder.

4.1 Parameter-efficient tuning for PMEs

To address the first cause of observed ineffective tuning, we reform the tuning method for PMEs. Instead of conducting full fine-tuning for all parameters, we propose tuning strategies specifically tailored to the message passing layers and embedding layers in PMEs, respectively.

(a) Vanilla MPP framework.   (b) Existing FSMPP framework.   (c) Our FSMPP framework.   (d) Our Pin-Tuning method for PMEs.
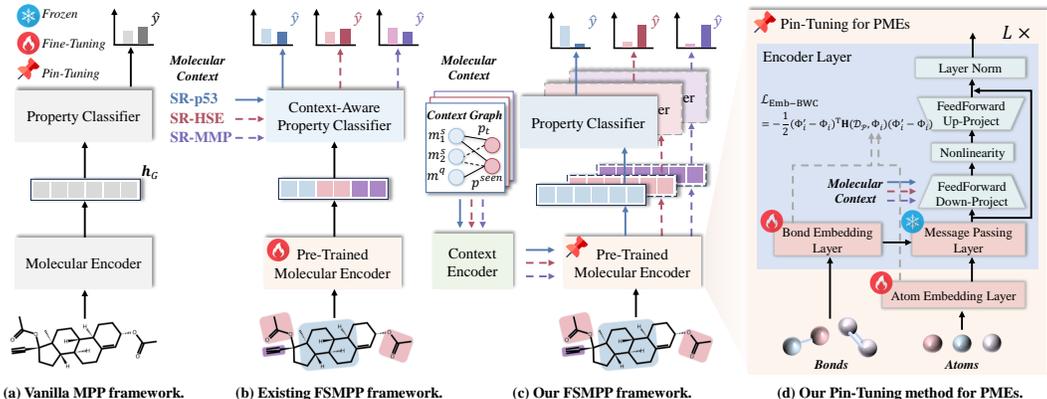
Figure 2: (a) The vanilla encoder-classifier framework for MPP. (b) The framework widely adopted by existing FSMPP methods, which contains a pre-trained molecular encoder and a context-aware property classifier. (c) Our proposed framework for FSMPP, in which we introduce a `Pin-Tuning` method to update the pre-trained molecular encoder followed by the property classifier. (d) The details of our proposed Pin-Tuning method for pre-trained molecular encoders. In (b) and (c), we use the property names like SR-HSE to denote the molecular context in episodes.

4.1.1 MP-Adapter: message passing layer-oriented adapter

For message passing layers in PMEs, the number of parameters is disproportionately large compared to the training samples. To mitigate this imbalance, we design a lightweight adapter targeted at the message passing layers, called `MP-Adapter`. The pre-trained parameters in each message passing layer include parameters in the MLP and the following batch normalization. We freeze all pre-trained parameters in message passing layers and add a lightweight trainable adapter after MLP in each message passing layer. Formally, the adapter module for l-th layer can be represented as:

$$z_v^{(l)} = \texttt{FeedForward}_{\texttt{down}}(h_v^{(l)}) \in \mathbb{R}^{d_2}, \tag{3}$$

$$\Delta h_v^{(l)} = \texttt{FeedForward}_{\texttt{up}}(\phi(z_v^{(l)})) \in \mathbb{R}^{d}, \tag{4}$$

$$\tilde{h}_v^{(l)} = \texttt{LayerNorm}(h_v^{(l)} + \Delta h_v^{(l)}) \in \mathbb{R}^{d}, \tag{5}$$

where $\texttt{FeedForward}(\cdot)$ denotes feed forward layer and $\texttt{LayerNorm}(\cdot)$ denotes layer normalization. To limit the number of parameters, we introduce a bottleneck architecture. The adapters downscale the original features from d dimensions to a smaller dimension d_2, apply nonlinearity ϕ, then upscale back to d dimensions. By setting d_2 smaller than d, we can limit the number of parameters added. The adapter module has a skip-connection internally. With the skip-connection, we adopt the near-zero initialization for parameters in the adapter modules, so that the modules are initialized to approximate identity functions. Therefore, the encoder with initialized adapters is equivalent to the pre-trained encoder. Furthermore, we add a layer normalization after skip-connection for training stability.

4.1.2 Emb-BWC: embedding layer-oriented Bayesian weight consolidation

Unlike message passing layers, embedding layers contain fewer parameters. Therefore, we directly fine-tune the parameters of the embedding layers, but impose a constraint to limit the magnitude of parameter updates, preventing aggressive optimization and catastrophic forgetting.

The parameters in an embedding layer consist of an embedding matrix used for lookups based on the indices of the original features. We stack the embedding matrices of all embedding layers to form $\Phi \in \mathbb{R}^{E \times d}$, where E represents the total number of lookup entries. Further, $\Phi_i \in \mathbb{R}^d$ denotes the i-th row's embedding vector, and $\Phi_{i,j} \in \mathbb{R}$ represents the j-th dimensional value of Φ_i.

To avoid aggressive optimization of Φ, we derive a Bayesian weight consolidation framework tailored for embedding layers, called `Emb-BWC`, by applying Bayesian learning theory [3] to fine-tuning.

Proposition 1: *(`Emb-BWC` ensures an appropriate stability-plasticity trade-off for pre-trained embedding layers.) Let $\Phi \in \mathbb{R}^{E \times d}$ be the pre-trained embeddings before fine-tuning, and $\Phi' \in \mathbb{R}^{E \times d}$ be the fine-tuned embeddings. Then, the embeddings can both retain the atom and bond properties*

obtained from pre-training and be appropriately updated to adapt to downstream FSMPP tasks, by introducing the following Emb-BWC loss into objective during the fine-tuning process:

$$\mathcal{L}_{\text{Emb-BWC}} = -\frac{1}{2}\sum_{i=1}^{E}(\Phi_i' - \Phi_i)^{\top}\mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i)(\Phi_i' - \Phi_i), \tag{6}$$

where $\mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i) \in \mathbb{R}^{d\times d}$ is the Hessian of the log likelihood $\mathcal{L}_{\mathcal{P}}$ of pre-training dataset $\mathcal{D}_{\mathcal{P}}$ at Φ_i.

Details on the theoretical derivation of Eq. (6) are given in Appendix A. Since $\mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i)$ is intractable to compute due to the great dimensionality of Φ, we adopt the diagonal approximation of Hessian. By approximating $\mathbf{H}$ as a diagonal matrix, the j-th value on the diagonal of $\mathbf{H}$ can be considered as the importance of the parameter $\Phi_{i,j}$. The following three choices are considered.

Identity matrix. When using the identity matrix to approximate the negation of $\mathbf{H}$, Eq. (6) is simplified to $\mathcal{L}_{\text{Emb-BWC}}^{\text{IM}} = \frac{1}{2}\sum_{i=1}^{E}\sum_{j=1}^{d}(\Phi_{i,j}' - \Phi_{i,j})^2$, assigning equal importance to each parameter. This loss function is also known as L2 penalty with pre-trained model as the starting point (L2-SP) [31].

Diagonal of Fisher information matrix. The Fisher information matrix (FIM) $\mathbf{F}$ is the negation of the expectation of the Hessian over the data distribution, i.e., $\mathbf{F} = -\mathbb{E}_{\mathcal{D}_{\mathcal{P}}}[\mathbf{H}]$, and the FIM can be further simplified with a diagonal approximation. Then, the Eq. (6) is simplified to $\mathcal{L}_{\text{Emb-BWC}}^{\text{FIM}} = \frac{1}{2}\sum_{i=1}^{E}\hat{\mathbf{F}}_i(\Phi_i' - \Phi_i)^2$, where $\hat{\mathbf{F}}_i \in \mathbb{R}^d$ is the diagonal of $\mathbf{F}(\mathcal{D}_{\mathcal{P}}, \Phi_i) \in \mathbb{R}^{d\times d}$ and the j-th value in $\hat{\mathbf{F}}_i$ is computed as $\mathbb{E}_{\mathcal{D}_{\mathcal{P}}}(\partial\mathcal{L}_{\mathcal{P}}/\partial\Phi_{i,j})^2$. This is equivalent to elastic weight consolidation (EWC) [24].

Diagonal of embedding-wise Fisher information matrix. In different property prediction tasks, the impact of the same atoms and inter-atomic interactions may be significant or negligible. Therefore, we propose this choice to assign importance to parameters based on different embeddings, rather than treating each parameter individually. By defining $\tilde{\Phi}_i = \sum_j \Phi_{i,j}$, the total update of the embedding Φ_i can be represented as $\Delta\Phi_i = \tilde{\Phi}_i' - \tilde{\Phi}_i = \sum_j(\Phi_{i,j}' - \Phi_{i,j})$. Then, the Eq. (6) is reformulated to $\mathcal{L}_{\text{Emb-EWC}}^{\text{EFIM}} = \frac{1}{2}\sum_{i=1}^{E}\tilde{\mathbf{F}}_i(\tilde{\Phi}_i' - \tilde{\Phi}_i)^2$, where $\tilde{\mathbf{F}}_i = \sum_j \mathbb{E}_{\mathcal{D}_{\mathcal{P}}}(\partial\mathcal{L}_{\mathcal{P}}/\partial\Phi_{i,j})^2$.

Detailed derivation is given in Appendix A. Intuitively, these three approximations employ different methods to assign importance to parameters. $\mathcal{L}_{\text{Emb-BWC}}^{\text{IM}}$ assigns the same importance to each parameter, $\mathcal{L}_{\text{Emb-BWC}}^{\text{FIM}}$ assigns individual importance to each parameter, and $\mathcal{L}_{\text{Emb-BWC}}^{\text{EFIM}}$ assigns the same importance to parameters within the same embedding vector.

4.2 Enabling contextual perceptiveness in MP-Adapter

For different property prediction tasks, the decisive substructures vary. As shown in Figure 2, the ester group in the given molecule determines the property SR-HSE, while the carbon-carbon triple bond determines the property SR-MMP. If fine-tuning can be guided by molecular context, encoding context-specific molecular representations allows for dynamic representations of molecules tailored to specific tasks and enables the modeling of the context-specific significance of substructures.

Extracting molecular context information. In each episode, we consider the labels of the support molecules on the target property and seen properties, as well as the labels of the query molecules on seen properties, as the context of this episode. We adopt the form of a graph to describe the context. Figure 3 demonstrates the transformation from original context data to a context graph. In the left table, the labels of molecules m_1^q, m_2^q for property p_t are the prediction targets, and the other shaded values are the available context. The right side shows the

Figure 3: Convert the context information of a 2-shot episode into a context graph.

context graph constructed based on the available context. Specifically, we construct context graph $\mathcal{G}_t = (\mathcal{V}_t, \mathbf{A}_t, \mathbf{X}_t)$ for episode E_t. It contains M molecule nodes $\{m\}$ and P property nodes $\{p\}$. Three types of edges indicate different relationships between molecules and properties.

Then we employ a GNN-based context encoder: $\mathbf{C} = \texttt{ContextEncoder}(\mathcal{V}_t, \mathbf{A}_t, \mathbf{X}_t)$, where $\mathbf{C} \in \mathbb{R}^{(M+P)\times d_2}$ denotes the learned context representation matrix for E_t. $\mathcal{V}_t$ and $\mathbf{A}_t$ denote the node set

and the adjacent matrix of the context graph, respectively, and $\mathbf{X}_t$ denotes the initial features of nodes. The features of molecule nodes are initialized with a pre-trained molecular encoder. The property nodes are randomly initialized. When we make the prediction of molecule m's target property p, we take the learned representations of the this molecule $\boldsymbol{c}_m$ and of the target property $\boldsymbol{c}_p$ as the context vectors. Details about the context encoder are provided in Appendix F.2.

In-context tuning with molecular context information. After obtaining the context vectors, we consider enabling the molecular encoder to use the context as a condition, achieving conditional molecular encoding. To achieve this, we further refine our adapter module. While neural conditional encoding has been explored in some domains, such as cross-attention [43] and ControlNet [68] for conditional image generation, these methods often come with a significant increase in the number of parameters. This contradicts our motivation of parameter-efficient tuning for few-shot tasks. In this work, we adopt a simple yet effective method. We directly concatenate the context with the output of the message passing layer, and feed them into the downscaling feed-forward layer in the `MP-Adapter`. Formally, the downscaling process defined in Eq. (3) is reformulated as:

$$z^{(l)} = \texttt{FeedForward}_{\texttt{down}}(h_v^{(l)} \| c_m \| c_p), \tag{7}$$

where $\|$ denotes concatenation. Such learned molecular representations are more easily predicted on specific properties, verified in Section 5.5 and Appendix G.

4.3 Optimization

Following MAML [10], a gradient descent strategy is adopted. Firstly, B episodes $\{E_t\}_{t=1}^B$ are randomly sampled. For each episode, in the inner-loop optimization, the loss on the support set is computed as $\mathcal{L}_{t,\mathcal{S}}^{cls}(f_\theta)$ and the parameters θ are updated by gradient descent:

$$\mathcal{L}_{t,\mathcal{S}}^{cls}(f_\theta) = -\sum\nolimits_{\mathcal{S}_t} (y \log(\hat{y}) + (1-y) \log(1-\hat{y})), \tag{8}$$

$$\theta' \leftarrow \theta - \alpha_{inner} \nabla_\theta \mathcal{L}_{t,\mathcal{S}}^{cls}(f_\theta), \tag{9}$$

where α_{inner} is the learning rate. In the outer loop, the classification loss of query set is denoted as $\mathcal{L}_{t,\mathcal{Q}}^{cls}$. Together with our `Emb-BWC` regularizer, the meta-training loss $\mathcal{L}(f_{\theta'})$ is computed and we do an outer-loop optimization with learning rate α_{outer} across the mini-batch:

$$\mathcal{L}(f_{\theta'}) = \frac{1}{B} \sum\nolimits_{t=1}^B \mathcal{L}_{t,\mathcal{Q}}^{cls}(f_{\theta'}) + \lambda \mathcal{L}_{\texttt{Emb-BWC}}, \tag{10}$$

$$\theta \leftarrow \theta - \alpha_{outer} \nabla_\theta \mathcal{L}(f_{\theta'}), \tag{11}$$

where λ is the weight of `Emb-BWC` regularizer. The pseudo-code is provided in Appendix B. We also provide more discussion of tunable parameter size and total model size in Appendix C.

5 Experiments

5.1 Evaluation setups

Datasets. We use five common few-shot molecular property prediction datasets from the MoleculeNet [61]: Tox21, SIDER, MUV, ToxCast, and PCBA. Standard data splits for FSMPP are adopted. Dataset statistics and more details of datasets can be found in Appendix D.

Baselines. For a comprehensive comparison, we adopt two types of baselines: (1) methods with molecular encoders trained from scratch, including Siamese Network [25], ProtoNet [46], MAML [10], TPN [35], EGNN [22], and IterRefLSTM [1]; and (2) methods which leverage pre-trained molecular encoders, including Pre-GNN [20], Meta-MGNN [14], PAR [58], and GS-Meta [73]. More details about these baselines are in Appendix E.

Metrics. Following prior works [1, 58], ROC-AUC scores are calculated on the query set for each meta-testing task, to evaluate the performance of FSMPP. We run experiments 10 times with different random seeds and report the mean and standard deviations.

7

Table 1: ROC-AUC scores (%) on benchmark datasets, compared with methods trained from scratch (first group) and methods that leverage pre-trained molecular encoder (second group). The best is marked with **boldface** and the second best is with underline. Δ*Improve.* indicates the relative improvements over the baseline models in percentage.

Model	Tox21 10-shot	Tox21 5-shot	SIDER 10-shot	SIDER 5-shot	MUV 10-shot	MUV 5-shot	ToxCast 10-shot	ToxCast 5-shot	PCBA 10-shot	PCBA 5-shot
Siamese	$80.40_{(0.35)}$	-	$71.10_{(4.32)}$	-	$59.96_{(5.13)}$	-	-	-	-	-
ProtoNet	$74.98_{(0.32)}$	$72.78_{(3.93)}$	$64.54_{(0.89)}$	$64.09_{(2.37)}$	$65.88_{(4.11)}$	$64.86_{(2.31)}$	$68.87_{(0.43)}$	$66.26_{(1.49)}$	$64.93_{(1.94)}$	$62.29_{(2.12)}$
MAML	$80.21_{(0.24)}$	$69.17_{(1.34)}$	$70.43_{(0.76)}$	$60.92_{(0.65)}$	$63.90_{(2.28)}$	$63.00_{(0.61)}$	$68.30_{(0.59)}$	$67.56_{(1.53)}$	$66.22_{(1.31)}$	$65.25_{(0.75)}$
TPN	$76.05_{(0.24)}$	$75.45_{(0.95)}$	$67.84_{(0.95)}$	$66.52_{(1.28)}$	$65.22_{(5.82)}$	$65.13_{(0.23)}$	$69.47_{(0.71)}$	$66.04_{(1.14)}$	$67.61_{(0.33)}$	$63.66_{(1.64)}$
EGNN	$81.21_{(0.16)}$	$76.80_{(2.62)}$	$72.87_{(0.73)}$	$60.61_{(1.06)}$	$65.20_{(2.08)}$	$63.46_{(2.58)}$	$74.02_{(1.11)}$	$67.13_{(0.50)}$	$69.92_{(1.85)}$	$67.71_{(3.67)}$
IterRefLSTM	$81.10_{(0.17)}$	-	$69.63_{(0.31)}$	-	$49.56_{(5.12)}$	-	-	-	-	-
Pre-GNN	$82.14_{(0.08)}$	$82.04_{(0.30)}$	$73.96_{(0.08)}$	$76.76_{(0.53)}$	$67.14_{(1.58)}$	$70.23_{(1.40)}$	$75.31_{(0.95)}$	$74.43_{(0.47)}$	$76.79_{(0.45)}$	$75.27_{(0.49)}$
Meta-MGNN	$82.97_{(0.10)}$	$76.12_{(0.23)}$	$75.43_{(0.21)}$	$66.60_{(0.38)}$	$68.99_{(1.84)}$	$64.07_{(0.56)}$	$76.27_{(0.56)}$	$75.26_{(0.43)}$	$72.58_{(0.34)}$	$72.51_{(0.52)}$
PAR	$84.93_{(0.11)}$	$83.95_{(0.15)}$	$78.08_{(0.16)}$	$77.70_{(0.34)}$	$\underline{69.96}_{(1.37)}$	$\underline{68.08}_{(2.42)}$	$79.41_{(0.08)}$	$76.89_{(0.32)}$	$73.71_{(0.61)}$	$72.79_{(0.98)}$
GS-Meta	$\underline{86.67}_{(0.41)}$	$\underline{86.43}_{(0.02)}$	$\underline{84.36}_{(0.54)}$	$\underline{84.57}_{(0.01)}$	$66.08_{(1.25)}$	$64.50_{(0.20)}$	$\underline{83.81}_{(0.16)}$	$\underline{82.65}_{(0.35)}$	$\underline{79.40}_{(0.43)}$	$\underline{77.47}_{(0.29)}$
Pin-Tuning	$\mathbf{91.56}_{(2.57)}$	$\mathbf{90.95}_{(2.33)}$	$\mathbf{93.41}_{(3.52)}$	$\mathbf{92.02}_{(3.01)}$	$\mathbf{73.33}_{(2.00)}$	$\mathbf{70.71}_{(1.42)}$	$\mathbf{84.94}_{(1.09)}$	$\mathbf{83.71}_{(0.93)}$	$\mathbf{81.26}_{(0.46)}$	$\mathbf{79.23}_{(0.52)}$
Δ*Improve.*	5.64%	5.23%	10.73%	8.81%	4.82%	3.86%	1.35%	1.28%	2.34%	2.27%

Table 2: Ablation analysis on the `MP-Adapter`, in which we drop different components to form variants. We report ROC-AUC scores (%), and the best performance is highlighted in **bold**.

Model	Adapter	Context	LayerNorm	Tox21 10-shot	Tox21 5-shot	SIDER 10-shot	SIDER 5-shot	MUV 10-shot	MUV 5-shot	ToxCast 10-shot	ToxCast 5-shot	PCBA 10-shot	PCBA 5-shot
Pin-Tuning	✓	✓	✓	**91.56**	**90.95**	**93.41**	**92.02**	**73.33**	**70.71**	**84.94**	**83.71**	**81.26**	**79.23**
w/o Adapter	-	-	✓	79.72	78.49	74.04	72.94	66.06	62.88	80.06	78.70	73.85	72.02
w/o Context	✓	-	✓	81.42	79.34	74.68	72.86	68.70	66.12	81.49	79.85	74.69	72.46
w/o LayerNorm	✓	✓	-	86.71	84.93	91.50	90.76	70.26	67.42	83.52	82.55	80.07	78.23

5.2 Performance comparison

We compare `Pin-Tuning` with the baselines and the results are summarized in Table 1, Table 7, and Table 8. Our method significantly outperforms all baseline models under both the 10-shot and 5-shot settings, demonstrating the effectiveness and superiority of our approach.

Across all datasets, our method provides greater improvement in the 10-shot scenario than in the 5-shot scenario. This is attributed to the molecular context constructed based on support molecules. When there are more molecules in the support set, the uncertainty in the context is reduced, providing more effective adaptation guidance for our parameter-efficient tuning.

Among benchmark datasets, our method shows significant improvement on the SIDER dataset, increasing by 10.73% in the 10-shot scenario and by 8.81% in the 5-shot scenario. We consider this is related to the relatively balanced ratio of positive to negative samples, as well as the absence of missing labels in the SIDER dataset (Table 5). A balanced and low-uncertainty distribution can better benefit addressing the FSMPP task from our method.

We also observe that the standard deviations of our method's results under 10 seeds are slightly higher than that of baseline models. However, our worst-case results are still better than the best baseline model. For example, in 10-shot experiments on the Tox21 dataset, the performance of our method is 91.56 ± 2.57. However, our 10 runs yield specific results with the worst-case ROC-AUC reaching 88.02, which is also better than the best baseline model GS-Meta's result of 86.67 ± 0.41. Therefore, a high standard deviation does not mean our method is inferior to baseline models.

5.3 Ablation study

For `MP-Adapter`, the main components consist of: (i) bottleneck adapter module (Adapter), (ii) introducing molecular context to adapters (Context), and (iii) layer normalization (LayerNorm). The results of ablation experiments are summarized in Table 2. The bottleneck adapter and the modeling of molecular context are the most critical, having the most significant impact on performance. Removing them leads to a noticeable decline, which underscores the importance of parameter-efficient tuning and context perceptiveness in FSMPP tasks. Layer normalization is used to normalize the resulting representations, which is also important for improving the optimization effect and stability.

For `Emb-BWC`, we verify the effectiveness of fine-tuning the embedding layers and regularizing them with different approximations of $\mathcal{L}_{\text{Emb-BWC}}$ (Table 3). Since the embedding layers have relatively few parameters, direct fine-tuning can also enhance performance. Applying our proposed regularizers to fine-tuning can further improve the effects. Among the three regularizers, the

Table 3: Ablation analysis on the `Emb-BWC`.

Fine-tune	Regularizer	Tox21	SIDER	MUV	PCBA
-	-	89.70	90.12	70.76	80.24
✓	-	90.17	<u>92.06</u>	72.37	80.74
✓	$\mathcal{L}_{\text{Emb-BWC}}^{\text{IM}}$	**91.56**	**93.41**	**73.22**	**81.26**
✓	$\mathcal{L}_{\text{Emb-BWC}}^{\text{FIM}}$	90.93	90.09	72.17	80.78
✓	$\mathcal{L}_{\text{Emb-BWC}}^{\text{EFIM}}$	<u>91.32</u>	90.31	<u>72.78</u>	<u>81.22</u>

$\mathcal{L}_{\text{Emb-BWC}}^{\text{IM}}$ is the most effective. This indicates that keeping pre-trained parameters to some extent can better utilize pre-trained knowledge, but the parameters worth keeping in fine-tuning and the important parameters in pre-training revealed by Fisher information matrix are not completely consistent.

5.4 Sensitivity analysis
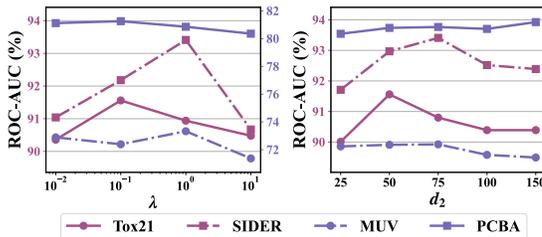
Figure 4: Effect of different hyper-parameters. The y-axis represents ROC-AUC scores (%) and the x-axis is the different hyper-parameters.
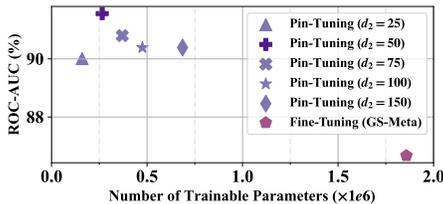
Figure 5: ROC-AUC (%) and number of trainable parameters of Pin-Tuning with varied value of d_2 and full Fine-Tuning method (e.g., GS-Meta) on the Tox21 dataset.

Effect of weight of `Emb-BWC` regularizer λ. `Emb-BWC` is applied on the embedding layers to limit the magnitude of parameter updates during fine-tuning. We vary the weight of this regularization λ from $\{0.01, 0.1, 1, 10\}$. The first subfigure in Figure 4 shows that the performance is best when $\lambda = 0.1$ or 1. When λ is too small, the parameters undergo too large updates on few-shot downstream datasets, leading to over-fitting and ineffectively utilizing the pre-trained knowledge. Too large λ causes the parameters of the embedding layers to be nearly frozen, which prevents effective adaptation.

Effect of hidden dimension of `MP-Adapter` d_2. The results corresponding to different values of d_2 from $\{25, 50, 75, 100, 150\}$ are presented in the second subfigure of Figure 4. On the Tox21 dataset, we further analyze the impact of this hyper-parameter on the number of trainable parameters. As shown in Figure 5, the number of parameters that our method needs to train is significantly less than that required by the full fine-tuning method, such as GS-Meta, while our method also performs better in terms of ROC-AUC performance due to solving over-fitting and context perceptiveness issues. When $d = 50$, Pin-Tuning performs best on Tox21, and the number of parameters that need to train is only 14.2% of that required by traditional fine-tuning methods.

5.5 Case study
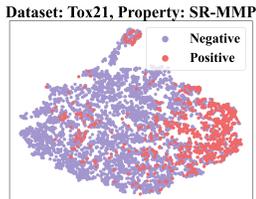
Figure 6: Molecular representations encoded by GS-Meta [73].
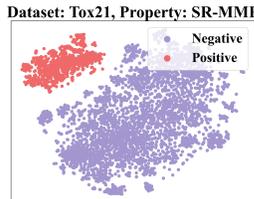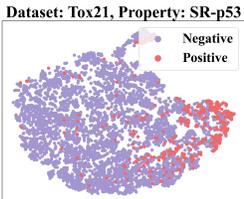
Figure 7: Molecular representations encoded by `Pin-Tuning`.

We visualized the molecular representations learned by the GS-Meta and our `Pin-Tuning`'s encoders in the 10-shot setting, respectively. As shown in Figure 6 and 7, `Pin-Tuning` can effectively adapt to different downstream tasks based on context information, generating property-specific molecular representations. Across different tasks, our method is more effective in encoding representations that

facilitate the prediction of the current property, reducing the difficulty of property prediction from the encoding representation aspect. More case studies are provided in Appendix G.

6 Conclusion

In this work, we propose a tuning method, `Pin-Tuning`, to address the ineffective fine-tuning of pre-trained molecular encoders in FSMPP tasks. Through the innovative parameter-efficient tuning and in-context tuning for pre-trained molecular encoders, our approach not only mitigates the issues of parameter-data imbalance but also enhances contextual perceptiveness. The promising results on public datasets underscore the potential of `Pin-Tuning` to advance this field, offering valuable insights for future research in drug discovery and material science.

Acknowledgments

References

[1] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS Central Science*, 3(4):283–293, 2017.

[2] Dingshuo Chen, Yanqiao Zhu, Jieyu Zhang, Yuanqi Du, Zhixun Li, Q. Liu, Shu Wu, and Liang Wang. Uncovering neural scaling laws in molecular representation learning. In *NeurIPS*, 2023.

[3] Haolin Chen and Philip N. Garner. Bayesian parameter-efficient fine-tuning for overcoming catastrophic forgetting. *arXiv*, abs/2402.12220, 2024.

[4] Wenlin Chen, Austin Tripp, and José Miguel Hernández-Lobato. Meta-learning adaptive deep kernel gaussian processes for molecular property prediction. In *ICLR*, 2023.

[5] Jianyuan Deng, Zhibo Yang, Hehe Wang, Iwao Ojima, Dimitris Samaras, and Fusheng Wang. A systematic study of key elements underlying molecular property prediction. *Nature Communications*, 14, 2023.

[6] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv*, abs/2203.06904, 2022.

[7] Ning Ding, Yujia Qin, Guang Yang, Fu Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Haitao Zheng, Jianfei Chen, Y. Liu, Jie Tang, Juanzi Li, and Maosong Sun. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5:220–235, 2023.

[8] Wei Dong, Dawei Yan, Zhijun Lin, and Peng Wang. Efficient adaptation of large vision transformer via adapter re-composing. 2023.

[9] Xiaomin Fang, Lihang Liu, Jieqiong Lei, Donglong He, Shanzhuo Zhang, Jingbo Zhou, Fan Wang, Hua Wu, and Haifeng Wang. Geometry-enhanced molecular representation learning for property prediction. *Nature Machine Intelligence*, 4:127 – 134, 2021.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[12] Jonathan Godwin, Michael Schaarschmidt, Alexander L. Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Velickovic, James Kirkpatrick, and Peter W. Battaglia. Simple GNN regularisation for 3d molecular property prediction and beyond. In *ICLR*, 2022.

[13] Renxiang Guan, Zihao Li, Wenxuan Tu, Jun Wang, Yue Liu, Xianju Li, Chang Tang, and Ruyi Feng. Contrastive multiview subspace clustering of hyperspectral images based on graph convolutional networks. *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[14] Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V. Chawla. Few-shot graph learning for molecular property prediction. In *WWW*, 2021.

[15] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, 2022.

[16] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:5149–5169, 2020.

[17] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *KDD*, 2022.

[18] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

[19] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[20] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.

[21] Dongki Kim, Jinheon Baek, and Sung Ju Hwang. Graph self-supervised learning with accurate discrepancy learning. In *NeurIPS*, 2022.

[22] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Edge-labeling graph neural network for few-shot learning. In *CVPR*, 2019.

[23] Suyeon Kim, Dongha Lee, SeongKu Kang, Seonghyeon Lee, and Hwanjo Yu. Learning topology-specific experts for molecular property prediction. In *AAAI*, pages 8291–8299, 2023.

[24] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114, 117.

[25] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015.

[26] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP (1)*, pages 3045–3059. Association for Computational Linguistics, 2021.

[27] Han Li, Dan Zhao, and Jianyang Zeng. KPGT: knowledge-guided pre-training of graph transformer for molecular property prediction. In *KDD*, pages 857–867, 2022.

[28] Shuangli Li, Jingbo Zhou, Tong Xu, Dejing Dou, and Hui Xiong. Geomgcl: Geometric graph contrastive learning for molecular property prediction. In *AAAI*, pages 4541–4549, 2022.

[29] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

[30] Xin Li, Dongze Lian, Zhihe Lu, Jiawang Bai, Zhibo Chen, and Xinchao Wang. Graphadapter: Tuning vision-language models with dual knowledge graph. In *NeurIPS*, 2023.

[31] Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *ICML*, 2018.

[32] Zhixun Li, Liang Wang, Xin Sun, Yifan Luo, Yanqiao Zhu, Dingshuo Chen, Yingtao Luo, Xiangxin Zhou, Qiang Liu, Shu Wu, Liang Wang, and Jeffrey Xu Yu. GSLB: the graph structure learning benchmark. In *NeurIPS*, 2023.

[33] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv*, abs/2303.15647, 2023.

[34] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *arXiv*, abs/2103.10385, 2021.

[35] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *ICLR*, 2019.

[36] Yue Liu, Jun Xia, Sihang Zhou, Siwei Wang, Xifeng Guo, Xihong Yang, Ke Liang, Wenxuan Tu, Stan Z. Li, and Xinwang Liu. A survey of deep graph clustering: Taxonomy, challenge, and application. *arXiv*, abs/2211.12875, 2022.

[37] Yue Liu, Ke Liang, Jun Xia, Sihang Zhou, Xihong Yang, Xinwang Liu, and Stan Z. Li. Dink-net: Neural clustering on large graphs. In *ICML*, 2023.

[38] Zhiyuan Liu, Yaorui Shi, An Zhang, Enzhi Zhang, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Rethinking tokenizer and decoder in masked graph modeling for molecules. In *NeurIPS*, 2023.

[39] Qiujie Lv, Guanxing Chen, Ziduo Yang, Weihe Zhong, and Calvin Yu-Chian Chen. Meta learning with graph attention networks for low-data drug discovery. *IEEE transactions on neural networks and learning systems*, 2023.

[40] Ziqiao Meng, Yaoman Li, Peilin Zhao, Yang Yu, and Irwin King. Meta-learning with motif-based task augmentation for few-shot molecular property prediction. In *SDM*, 2023.

[41] Cuong Q. Nguyen, Constantine Kreatsoulas, and Kim Branson. Meta-learning gnn initializations for low-resource molecular property prediction. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[42] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL*, 2021.

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10674–10685. IEEE, 2022.

[44] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *NeurIPS*, 2020.

[45] Johannes Schimunek, Philipp Seidl, Lukas Friedrich, Daniel Kuhn, Friedrich Rippmann, Sepp Hochreiter, and Günter Klambauer. Context-enriched molecule representations improve few-shot drug discovery. In *ICLR*, 2023.

[46] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[47] Ying Song, Shuangjia Zheng, Zhangming Niu, Zhang-Hua Fu, Yutong Lu, and Yuedong Yang. Communicative representation learning on attributed molecular graphs. In *IJCAI*, 2020.

[48] Yisheng Song, Ting Wang, Puyu Cai, Subrota K. Mondal, and Jyoti Prakash Sahoo. A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *ACM Computing Surveys*, 2023.

[49] Megan Stanley, John Bronskill, Krzysztof Maziarz, Hubert Misztela, Jessica Lanini, Marwin H. S. Segler, Nadine Schneider, and Marc Brockschmidt. Fs-mol: A few-shot learning dataset of molecules. In *NeurIPS Datasets and Benchmarks*, 2021.

[50] Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Lió. 3d infomax improves gnns for molecular property prediction. In *ICML*, pages 20479–20502, 2022.

[51] Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. Mocl: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *KDD*, 2021.

[52] Xin Sun, Liang Wang, Qiang Liu, Shu Wu, Zilei Wang, and Liang Wang. DIVE: subgraph disagreement for graph out-of-distribution generalization. In *KDD*, 2024.

[53] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, page 3637–3645, 2016.

[54] Liang Wang, Xiang Tao, Qiang Liu, Shu Wu, and Liang Wang. Rethinking graph masked autoencoders through alignment and uniformity. In *AAAI*, 2024.

[55] Liang Wang, Shu Wu, Q. Liu, Yanqiao Zhu, Xiang Tao, and Mengdi Zhang. Bi-level graph structure learning for next poi recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 36:5695–5708, 2024.

[56] Xu Wang, Huan Zhao, Wei-Wei Tu, and Quanming Yao. Automated 3d pre-training for molecular property prediction. In *KDD*, pages 2419–2430, 2023.

[57] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 2020.

[58] Yaqing Wang, Abulikemu Abuduweili, Quanming Yao, and Dejing Dou. Property-aware relation networks for few-shot molecular property prediction. In *NeurIPS*, 2021.

[59] Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *EMNLP*, 2022.

[60] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 2022.

[61] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet: A benchmark for molecular machine learning. *arXiv*, abs/1703.00564, 2017.

[62] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-bert: Rethinking pre-training graph neural networks for molecules. In *ICLR*, 2023.

[63] Yi Xiao, Xiangxin Zhou, Qiang Liu, and Liang Wang. Bridging text and molecule: A survey on multimodal frameworks for molecule. *arXiv*, abs/2403.13830, 2024.

[64] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *TPAMI*, 2022.

[65] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[66] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *NeurIPS*, 2021.

[67] Sheheryar Zaidi, Michael Schaarschmidt, James Martens, Hyunjik Kim, Yee Whye Teh, Alvaro Sanchez-Gonzalez, Peter W. Battaglia, Razvan Pascanu, and Jonathan Godwin. Pre-training via denoising for molecular property prediction. In *ICLR*, 2023.

[68] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, pages 3813–3824. IEEE, 2023.

[69] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*, 2023.

[70] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Jiao Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. In *ICLR*, 2024.

[71] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. Motif-based graph self-supervised learning for molecular property prediction. In *NeurIPS*, pages 15870–15882, 2021.

[72] Yanqiao Zhu, Dingshuo Chen, Yuanqi Du, Yingze Wang, Q. Liu, and Shu Wu. Molecular contrastive pretraining with collaborative featurizations. *Journal of Chemical Information and Modeling*, 2024.

[73] Xiang Zhuang, Qiang Zhang, Bin Wu, Keyan Ding, Yin Fang, and Huajun Chen. Graph sampling-based meta-learning for molecular property prediction. In *IJCAI*, 2023.

Appendix

The organization of the appendix is as follows:

- Appendix A: Derivation of `Emb-BWC` regularization;
- Appendix B: Pseudo-code of training process;
- Appendix C: Discussion of tunable parameter size and total model size;
- Appendix D: Details of datasets;
- Appendix E: Details of baselines;
- Appendix F: Implementation details;
- Appendix G: More experimental results and discussions;
- Appendix H: Limitations and future directions.

A Derivation of `Emb-BWC` regularization

A.1 Derivation of $\mathcal{L}_{\text{Emb-BWC}}$

Let $\Phi \in \mathbb{R}^{E \times d}$ be the pre-trained embeddings before fine-tuning, and $\Phi' \in \mathbb{R}^{E \times d}$ be the fine-tuned embeddings. Further, $\Phi_i \in \mathbb{R}^d$ denotes the i-th row's embedding vector in Φ, and $\Phi_{i,j} \in \mathbb{R}$ represents the j-th dimensional value of Φ_i.

The optimization of embedding layers can be interpreted as performing a maximum a posterior (MAP) estimation of the parameters Φ' given the pre-training data and training data of downstream FSMPP task, which is formulated in a Bayesian framework.

In the FSMPP setting, the molecular encoder has been pre-trained on the pre-training task $\mathcal{P}$ using data $\mathcal{D}_{\mathcal{P}}$, and is then fine-tuned on a downstream FSMPP task $\mathcal{F}$ using data $\mathcal{D}_{\mathcal{F}}$. The overall objective is to find the optimal parameters on task $\mathcal{F}$ while preserving the prior knowledge obtained in pre-training on task $\mathcal{F}$. Based on a prior $p(\Phi')$ of the embedding parameters, the posterior after observing the FSMPP task $\mathcal{F}$ can be computed with Bayes' rule:

$$
\begin{aligned}
p(\Phi'|\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{F}}) &= \frac{p(\mathcal{D}_{\mathcal{F}}|\Phi', \mathcal{D}_{\mathcal{P}})p(\Phi'|\mathcal{D}_{\mathcal{P}})}{p(\mathcal{D}_{\mathcal{F}}|\mathcal{D}_{\mathcal{P}})} \\
&= \frac{p(\mathcal{D}_{\mathcal{F}}|\Phi')p(\Phi'|\mathcal{D}_{\mathcal{P}})}{p(\mathcal{D}_{\mathcal{F}})},
\end{aligned}
\tag{12}
$$

where $\mathcal{D}_{\mathcal{F}}$ is assumed to be independent of $\mathcal{D}_{\mathcal{P}}$. Taking a logarithm of the posterior, the MAP objective is therefore:

$$
\begin{aligned}
\Phi'^* &= \arg\max_{\Phi'} \log p(\Phi'|\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{F}}) \\
&= \arg\max_{\Phi'} \log p(\mathcal{D}_{\mathcal{F}}|\Phi') + \log p(\Phi'|\mathcal{D}_{\mathcal{P}}).
\end{aligned}
\tag{13}
$$

The first term $\log p(\mathcal{D}_{\mathcal{F}}|\Phi')$ is the log likelihood of the data $\mathcal{D}_{\mathcal{F}}$ given the parameters Φ', which can be expressed as the training loss function on task $\mathcal{F} = -\log p(\mathcal{D}_{\mathcal{F}}|\Phi')$, denoted as $\mathcal{L}_{\mathcal{F}}(\Phi')$. The second term $p(\Phi'|\mathcal{D}_{\mathcal{P}})$ is the posterior of the parameters given the pre-training dataset $\mathcal{D}_{\mathcal{P}}$. Since $\Phi' = [\Phi_1'^\top, \Phi_2'^\top, \ldots, \Phi_E'^\top]^\top$, and Φ_i' is conditionally independent of Φ_j' for $i, j = \{1, \ldots, E\}$ and $i \neq j$ given condition $\mathcal{D}_{\mathcal{P}}$, we have $p(\Phi'|\mathcal{D}_{\mathcal{P}}) = \prod_{i=1}^{E} p(\Phi_i'|\mathcal{D}_{\mathcal{P}})$. Thus, $\log p(\Phi'|\mathcal{D}_{\mathcal{P}}) = \sum_{i=1}^{E} \log p(\Phi_i'|\mathcal{D}_{\mathcal{P}})$

For adapting pre-trained molecular embedding layers to downstream FMSPP tasks, this posterior must encompass the prior knowledge of the pre-trained embedding layers to reflect which parameters are important for pre-training task $\mathcal{P}$. Despite the true posterior being intractable, $\log p(\Phi_i'|\mathcal{D}_{\mathcal{P}})$ can be defined as a function $f(\Phi_i')$ and approximated around the optimum point $f(\Phi_i)$, where $f(\Phi_i)$ is the pre-trained values and $\nabla f(\Phi_i) = 0$. Performing a second-order Taylor expansion on $f(\Phi_i')$ around Φ_i gives:

$$
\begin{aligned}
\log p\left(\Phi_i' \mid \mathcal{D}_{\mathcal{P}}\right) &\approx f\left(\Phi_i\right) + \frac{1}{2}\left(\Phi_i' - \Phi_i\right)^T \nabla^2 f\left(\Phi_i\right)\left(\Phi_i' - \Phi_i\right) \\
&= f\left(\Phi_i\right) + \frac{1}{2}\left(\Phi_i' - \Phi_i\right)^T \mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i)\left(\Phi_i' - \Phi_i\right),
\end{aligned}
\tag{14}
$$

where $\mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i) \in \mathbb{R}^{d \times d}$ is the Hessian matrix of $f(\Phi_i')$ at Φ_i. The second term suggests that the posterior of the parameters on the pre-training data can be approximated by a Gaussian distribution with mean Φ_i' and covariance $\mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i)^{-1}$. Following Eq. (13), the training objective becomes:

$$
\begin{aligned}
\Phi'^* &= \arg\max_{\Phi'} \log p(\mathcal{D}_{\mathcal{F}}|\Phi') + \log p(\Phi'|\mathcal{D}_{\mathcal{P}}) \\
&= \arg\max_{\Phi'} \log p(\mathcal{D}_{\mathcal{F}}|\Phi') + \sum_{i=1}^{E} \log p(\Phi_i'|\mathcal{D}_{\mathcal{P}}) \\
&= \arg\min_{\Phi'} \mathcal{L}_{\mathcal{F}}(\Phi') - \sum_{i=1}^{E} f(\Phi_i) - \frac{1}{2} \sum_{i=1}^{E} (\Phi_i' - \Phi_i)^T \mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i) (\Phi_i' - \Phi_i) \\
&= \arg\min_{\Phi'} \mathcal{L}_{\mathcal{F}}(\Phi') - \frac{1}{2} \sum_{i=1}^{E} (\Phi_i' - \Phi_i)^T \mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i) (\Phi_i' - \Phi_i).
\end{aligned}
\tag{15}
$$

We define the second term as our `Emb-BWC` regularization objective:

$$
\mathcal{L}_{\text{Emb-BWC}} = -\frac{1}{2} \sum_{i=1}^{E} (\Phi_i' - \Phi_i)^\top \mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i)(\Phi_i' - \Phi_i).
\tag{16}
$$

A.2 Derivation of $\mathcal{L}_{\text{Emb-BWC}}^{\text{FIM}}$

Since the Fisher information matrix (FIM) $\mathbf{F}$ is the negation of the expectation of the Hessian over the data distribution, i.e., $\mathbf{F} = -\mathbb{E}_{\mathcal{D}_{\mathcal{P}}}[\mathbf{H}]$, the objective can be reformulated as:

$$
\mathcal{L}_{\text{Emb-BWC}}^{\text{FIM}} = \frac{1}{2} \sum_{i=1}^{E} (\Phi_i' - \Phi_i)^\top \mathbf{F}(\mathcal{D}_{\mathcal{P}}, \Phi_i)(\Phi_i' - \Phi_i),
\tag{17}
$$

where $\mathbf{F}(\mathcal{D}_{\mathcal{P}}, \Phi_i) \in \mathbb{R}^{d \times d}$ is the corresponding Fisher information matrix of $\mathbf{H}(\mathcal{D}_{\mathcal{P}}, \Phi_i)$. Further, the Fisher information matrix can be further simplified with a diagonal approximation. Then, the objective is simplified to:

$$
\mathcal{L}_{\text{Emb-BWC}}^{\text{FIM}} \approx \frac{1}{2} \sum_{i=1}^{E} \hat{\mathbf{F}}_i (\Phi_i' - \Phi_i)^2,
\tag{18}
$$

where $\hat{\mathbf{F}}_i \in \mathbb{R}^d$ is the diagonal of $\mathbf{F}(\mathcal{D}_{\mathcal{P}}, \Phi_i)$. According to the definition of the Fisher information matrix, the j-th value in $\hat{\mathbf{F}}_i$ is computed as $\mathbb{E}_{\mathcal{D}_{\mathcal{P}}}(\partial \mathcal{L}_{\mathcal{P}}/\partial \Phi_{i,j})^2$. In this work, this approximated form is defined as $\mathcal{L}_{\text{Emb-BWC}}^{\text{FIM}}$.

A.3 Derivation of $\mathcal{L}_{\text{Emb-BWC}}^{\text{EFIM}}$

We assume that the parameters within an embedding should share the same importance. To this end, we define $\tilde{\Phi}_i = \sum_j \Phi_{i,j}$, then the total update of the embedding Φ_i can be represented as $\Delta \Phi_i = \tilde{\Phi}_i' - \tilde{\Phi}_i = \sum_j (\Phi_{i,j}' - \Phi_{i,j})$. Then, the objective in Eq. (16) is reformulated to:

$$
\mathcal{L}_{\text{Emb-EWC}}^{\text{EFIM}} = \frac{1}{2} \sum_{i=1}^{E} \tilde{\mathbf{H}}_i (\tilde{\Phi}_i' - \tilde{\Phi}_i)^2,
\tag{19}
$$

where $\tilde{\mathbf{H}}_i = \frac{\partial^2 \mathcal{L}_{\mathcal{P}}}{\partial \tilde{\Phi}_i^2} = \frac{\partial}{\partial \tilde{\Phi}_i}(\frac{\partial \mathcal{L}_{\mathcal{P}}}{\partial \tilde{\Phi}_i})$. Next, we continue to derive $\tilde{\mathbf{H}}_i$. Given that $\tilde{\Phi}_i = \sum_{j=1}^{d} \Phi_{i,j}$, we first use the chain rule to find $\frac{\partial \mathcal{L}_{\mathcal{P}}}{\partial \tilde{\Phi}_i}$. According to chain rule, the derivative of $\mathcal{L}_{\mathcal{P}}$ with respect to $\partial \tilde{\Phi}_i$ can be computed as:

$$
\frac{\partial \mathcal{L}_{\mathcal{P}}}{\partial \tilde{\Phi}_i} = \sum_j \frac{\partial \mathcal{L}_{\mathcal{P}}}{\partial \Phi_{i,j}} \frac{\partial \Phi_{i,j}}{\partial \tilde{\Phi}_i}.
\tag{20}
$$

Since $\tilde{\Phi}_i = \Phi_{i,1} + \Phi_{i,2} + \ldots + \Phi_{i,d}$, each of $\frac{\partial \Phi_{i,j}}{\partial \tilde{\Phi}_i}$ for $j = 1, 2, \ldots, d$ equals 1. Therefore, the equation simplifies to:

$$
\frac{\partial \mathcal{L}_{\mathcal{P}}}{\partial \tilde{\Phi}_i} = \sum_{j=1}^{d} \frac{\partial \mathcal{L}_{\mathcal{P}}}{\partial \Phi_{i,j}}.
\tag{21}
$$

When taking the derivative of this with respect to $\Phi_{i,j}$ again, using the chain rule, we have:

$$\frac{\partial^2 \mathcal{L}_\mathcal{P}}{\partial \tilde{\Phi}_i^2} = \frac{\partial}{\partial \tilde{\Phi}_i}(\sum_{j=1}^{d} \frac{\partial \mathcal{L}_\mathcal{P}}{\partial \Phi_{i,j}}) = \sum_{j=1}^{d}\sum_{k=1}^{d} \frac{\partial}{\partial \Phi_{i,k}}(\frac{\partial \mathcal{L}_\mathcal{P}}{\partial \Phi_{i,j}})\frac{\partial \Phi_{i,k}}{\partial \tilde{\Phi}_i} \tag{22}$$

Given that $\Phi_{i,j}(\Phi_{i,k})$ are all parameters in embedding lookup tables, they are independent of each other. Thus, when $j = k$, $\frac{\partial}{\partial \Phi_{i,k}}(\frac{\partial \mathcal{L}_\mathcal{P}}{\partial \Phi_{i,j}})\frac{\partial \Phi_{i,k}}{\partial \tilde{\Phi}_i} = \frac{\partial^2 \mathcal{L}_\mathcal{P}}{\partial \Phi_{i,j}^2}$, otherwise it equals 0. We finally get $\tilde{\mathbf{H}}_i = \sum_j \frac{\partial^2 \mathcal{L}_\mathcal{P}}{\partial \Phi_{i,j}^2} = \sum_j \mathbf{H}(\mathcal{D}_\mathcal{P}, \Phi_i)_{j,j}$.

We still approximate the Hessian with the FIM as in Section A.2, and combining this with the definition of FIM, we arrive at the final objective:

$$\mathcal{L}_{\text{Emb-EWC}}^{\text{EFIM}} \approx \frac{1}{2}\sum_{i=1}^{E} \tilde{\mathbf{F}}_i(\tilde{\Phi}_i' - \tilde{\Phi}_i)^2, \tag{23}$$

where $\tilde{\mathbf{F}}_i = \sum_j \mathbb{E}_{\mathcal{D}_\mathcal{P}}(\partial \mathcal{L}_\mathcal{P}/\partial \Phi_{i,j})^2$.

B Pseudo-code of training process

To help better understand the training process, we provide the brief pseudo-code of it in Algorithm 1.

Algorithm 1: Training process of `Pin-Tuning`.

Input : Training set $\mathcal{D}_{\text{train}}$
Output : Tuned few-shot molecular property prediction model with parameter θ

1 **while** *not converge* **do**
2   Sample B episode from training set $\mathcal{D}_{\text{train}}$ to form a mini-batch $\{E_t\}_{t=1}^{B}$;
3   **for** $t = 1$ **to** B **do**
4   Calculate classification loss on support set $\mathcal{L}_{t,\mathcal{S}}^{cls}(f_\theta)$ by Eq. (8) on E_t:
        $\theta' \leftarrow \theta - \alpha_{inner}\nabla_\theta \mathcal{L}_{t,\mathcal{S}}^{cls}(f_\theta)$;
5   Do inner-loop update by Eq. (9) on E_t;
6   Calculate classification loss on query set $\mathcal{L}_{t,\mathcal{Q}}^{cls}(f_{\theta'})$ by Eq. (8) on E_t;
7   Calculate update constraint $\mathcal{L}_{\text{Emb-BWC}}$ by Eq. (6);
8   Do outer-loop optimization by Eq. (10) and Eq. (11): $\theta \leftarrow \theta - \alpha_{outer}\nabla_\theta \mathcal{L}(f_{\theta'})$;
9 Return optimized model parameter θ.

C Discussion of tunable parameter size and total model size

C.1 Tunable parameter size of molecular encoder

We compare the tunable parameter size of full fine-tuning and our `Pin-Tuning`. Section 3.3 describes the parameters of the PME, which include those for the embedding layers and the message passing layers. We assume there are $|E_n|$ original node features and $|E_e|$ edge features. Considering there is one node embedding layer and L edge embedding layers, the total number of parameters for the embedding part is $|E_n|d + L|E_e|d$. The parameters in the message passing layer consist of the 2-layer MLP including biases shown in Eq. (2) and its subsequent batch normalization, with each layer having $L(2dd_1 + d + d_1 + 2d)$ parameters. In summary, the total number of parameters to update in full fine-tuning is

$$N_{Fine-Tuning} = |E_n|d + L(|E_e|d + 2dd_1 + 3d + d_1). \tag{24}$$

In our `Pin-Tuning` method, the parameters of the embedding layers are still updated. However, in each message passing layer, the original parameters are completely frozen, and the parts that require updating are the two feed-forward layers and the layer normalization in the bottleneck adapter

Table 4: Comparison of total model size. * indicates that the parameters are frozen.

	GS-Meta	Ours
Size of Molecular Encoder	1.86M	1.86M*
Size of Adapter	-	0.21M
Size of Context Encoder	0.62M	0.62M
Size of Classifier	0.18M	0.27M
Size of Total Model	2.66M	2.96M
Size of Tunable Part of the Model	2.66M	1.10M

Table 5: Dataset statistics.

Dataset	Tox21	SIDER	MUV	ToxCast	PCBA
#Compound	7831	1427	93127	8575	437929
#Property	12	27	17	617	128
#Train Property	9	21	12	451	118
#Test Property	3	6	5	158	10
%Positive Label	6.24	56.76	0.31	12.60	0.84
%Negative Label	76.71	43.24	15.76	72.43	59.84
%Unknown Label	17.05	0	84.21	14.97	39.32

module, amounting to $L(2dd_2 + d + d_2 + 2d)$ parameters for this part. Therefore, the total number of parameters that need to be updated in our `Pin-Tuning` is

$$N_{Pin-Tuning} = |E_n|d + L(|E_e|d + 2dd_2 + 3d + d_2). \tag{25}$$

The difference in the number of parameters updated between the two tuning methods is $\Delta N = (d_1 - d_2)L(2d + 1)$.

C.2   Total model size

We provide a comparison of total model size between our `Pin-Tuning` and the state-of-the-art baseline method, GS-Meta. The total model size consists of both frozen parameters and trainable parameters. The results are presented in Table 4. The total size of our model is comparable to GS-Meta, but the number of parameters that need to be trained is far less than GS-Meta.

D   Details of datasets

We carry out experiments in MoleculeNet benchmark [61] on five widely used few-shot molecular property prediction datasets:

- **Tox21**: This dataset covers qualitative toxicity measurements and was utilized in the 2014 Tox21 Data Challenge.
- **SIDER**: The Side Effect Resource (SIDER) functions as a repository for marketed drugs and adverse drug reactions (ADR), categorized into 27 system organ classes.
- **MUV**: The Maximum Unbiased Validation (MUV) is determined through the application of a refined nearest neighbor analysis, specifically designed for validating virtual screening techniques.
- **ToxCast**: This dataset comprises a compilation of compounds with associated toxicity labels derived from high-throughput screening.
- **PCBA**: PubChem BioAssay (PCBA) represents a database containing the biological activities of small molecules generated through high-throughput screening.

Dataset statistics are summarized in Table 5 and Table 6.

E   Details of baselines

We compare our `Pin-Tuning` with two types of baseline models for few-shot molecular property prediction tasks, categorized according to the training strategy of molecular encoders: trained-from-scratch methods and pre-trained methods.

Table 6: Statistics of sub-datasets of ToxCast.

Assay Provider	#Compound	#Property	#Train Property	#Test Property	%Label active	%Label inactive	%Missing Label
APR	1039	43	33	10	10.30	61.61	28.09
ATG	3423	146	106	40	5.92	93.92	0.16
BSK	1445	115	84	31	17.71	82.29	0.00
CEETOX	508	14	10	4	22.26	76.38	1.36
CLD	305	19	14	5	30.72	68.30	0.98
NVS	2130	139	100	39	3.21	4.52	92.27
OT	1782	15	11	4	9.78	87.78	2.44
TOX21	8241	100	80	20	5.39	86.26	8.35
Tanguay	1039	18	13	5	8.05	90.84	1.11

Trained-from-scratch methods:

- **Siamese** [25]: Siamese is used to rank similarity between input molecule pairs with a dual network.

- **ProtoNet** [46]: ProtoNet learns a metric space for few-shot classification, enabling classification by calculating the distances between each query molecule and the prototype representations of each class.

- **MAML** [10]: MAML adapts the meta-learned parameters to achieve good generalization performance on new tasks with a small amount of training data and gradient steps.

- **TPN** [35]: TPN classifies the entire test set at once by learning to propagate labels from labeled instances to unlabeled test instances using a graph construction module that exploits the manifold structure in the data.

- **EGNN** [22]: EGNN predicts edge labels on a graph constructed from input samples to explicitly capture intra-cluster similarity and inter-cluster dissimilarity.

- **IterRefLSTM** [1]: IterRefLSTM adapts Matching Networks [53] to handle molecular property prediction tasks.

Pre-trained methods:

- **Pre-GNN** [20]: Pre-GNN is a classic pre-trained molecular model, taking the GIN as backbone and pre-training it with different self-supervised tasks.

- **Meta-MGNN** [14]: Meta-MGNN leverages Pre-GNN for learning molecular representations and incorporates meta-learning and self-supervised learning techniques.

- **PAR** [58]: PAR uses class prototypes to update input representations and designs label propagation for similar inputs in the relational graph, thus enabling the transformation of generic molecular embeddings into property-aware spaces.

- **GS-Meta** [73]: GS-Meta constructs a Molecule-Property relation graph (MPG) and redefines episodes in meta-learning as subgraphs of the MPG.

Following prior work [58], for the methods we reproduced, we use GIN as the graph-based molecular encoder to extract molecular representations. Specifically, we use the GIN provided by Pre-GNN [20] which consists of 5 GIN layers with 300-dimensional hidden units. Pre-GNN, Meta-MGNN, PAR, and GS-Meta further use the pre-trained GIN which is also provided by Pre-GNN.

F   Implementation details

F.1   Hardware and software

Our experiments are conducted on Linux servers equipped with an AMD CPU EPYC 7742 (256) @ 2.250GHz, 256GB RAM and NVIDIA 3090 GPUs. Our model is implemented in PyTorch version 1.12.1, PyTorch Geometric version 2.3.1 (https://pyg.org/) with CUDA version 11.3, RDKit version 2023.3.3 and Python 3.9.18. Our code is available at: `https://github.com/CRIPAC-DIG/Pin-Tuning`.

F.2   Model configuration

For featurization of molecules, we use atomic number and chirality tag as original atom features, as well as bond type and bond direction as bond features, which is in line with most molecular

Table 7: 10-shot performance on each sub-dataset of ToxCast.

Model	APR	ATG	BSK	CEETOX	CLD	NVS	OT	TOX21	Tanguay
ProtoNet	73.58	59.26	70.15	66.12	78.12	65.85	64.90	68.26	73.61
MAML	72.66	62.09	66.42	64.08	74.57	66.56	64.07	68.04	77.12
TPN	74.53	60.74	65.19	66.63	75.22	63.20	64.63	73.30	81.75
EGNN	80.33	66.17	73.43	66.51	78.85	71.05	68.21	76.40	85.23
Pre-GNN	80.61	67.59	76.65	66.52	78.88	75.09	70.52	77.92	83.05
Meta-MGNN	81.47	69.20	78.97	66.57	78.30	79.60	69.55	78.77	83.98
PAR	86.09	72.72	82.45	72.12	83.43	74.94	71.96	82.81	88.20
GS-Meta	90.15	82.54	88.21	74.19	86.34	76.29	74.47	90.63	91.47
Pin-Tuning	**92.78**	**83.58**	**89.49**	**75.96**	**87.70**	**76.33**	**75.56**	**90.80**	**92.25**
Δ*Improve.*	2.92%	1.26%	1.45%	2.39%	1.58%	0.05%	1.46%	0.19%	0.85%

Table 8: 5-shot performance on each sub-dataset of ToxCast.

Model	APR	ATG	BSK	CEETOX	CLD	NVS	OT	TOX21	Tanguay
ProtoNet	70.38	58.11	63.96	63.41	76.70	62.27	64.52	65.99	70.98
MAML	68.88	60.01	67.05	62.42	73.32	69.18	64.56	66.73	75.88
TPN	70.76	57.92	63.41	64.73	70.44	61.36	61.99	66.49	77.27
EGNN	74.06	60.56	64.60	63.20	71.44	62.62	66.70	65.33	75.69
Pre-GNN	80.38	66.96	75.64	64.88	78.03	74.08	70.42	75.74	82.73
Meta-MGNN	81.22	69.90	79.67	65.78	77.53	73.99	69.20	76.25	83.76
PAR	83.76	70.24	80.82	69.51	81.32	70.60	71.31	79.71	84.71
GS-Meta	89.36	81.92	86.12	74.48	83.10	74.72	73.26	89.71	91.15
Pin-Tuning	**89.94**	**82.37**	**87.61**	**75.20**	**85.07**	**75.49**	**74.70**	**90.89**	**92.14**
Δ*Improve.*	0.65%	0.55%	1.73%	0.97%	2.37%	1.03%	1.97%	1.32%	1.09%

pre-training methods. Following previous works, we set $d = 300$. For MLPs in Eq. (2), we use the ReLU activation with $d_1 = 600$. Pre-trained GIN model provided by Pre-GNN [20] is adopted as the PTME in our framework. We tune the weight of update constraint (i.e., λ) in {0.01, 0.1, 1, 10}, tune the learning rate of inner loop (i.e., α_{inner}) in {1e-3, 5e-3, 1e-2,5e-2,1e-1, 5e-1, 1, 5}, and tune the learning rate of outer loop (i.e., α_{outer}) in {1e-5, 1e-4, 1e-3,1e-2,1e-1}. Based on the results of hyper-parameter tuning, we adopt $\alpha_{inner} = 0.5, \alpha_{outer} = 1e - 3$ and $d_2 = 50$. The ContextEncoder($\cdot$) described in Section 4.2 is implemented using a 2-layer message passing neural network [11]. In each MPNN layer, we employ a linear layer to aggregate messages from the neighborhoods of nodes and utilize distinct edge features to differentiate between various edge types in the context graphs. For baselines, we follow their recommended settings.

G More experimental results and discussions

More discussion of Figure 1. The results show that molecular encoders with more molecule-specific inductive biases, such as CMPNN [47] and Graphormer [66], performed slightly worse than GIN-Mol [20] on this few-shot task. This is because more complex encoders require more parameters to provide inductive biases, which are difficult to train effectively under a few-shot setting.

More main results. The detailed comparison between Pin-Tuning and baseline models on sub-datasets of ToxCast are summarized in Table 7 and Table 8. Our method outperforms all baseline models under both the 10-shot and 5-shot settings, demonstrating the superiority of our method compared to existing methods.

More discussion of ablation study. Different datasets show varying sensitivity to the removal of components. On small-scale datasets like Tox21 and SIDER, removing components leads to a significant performance drop. On large-scale datasets like ToxCast and PCBA, the impact of removing components is less pronounced. This is because more episodes can be constructed on large-scale datasets, which aids in adaptation. This observation indicates that Pin-Tuning can bring considerable benefits in situations where data is extremely scarce.

More case studies. We provide more case studies in Figure 8 and 9 as a supplement to Section 5.5.
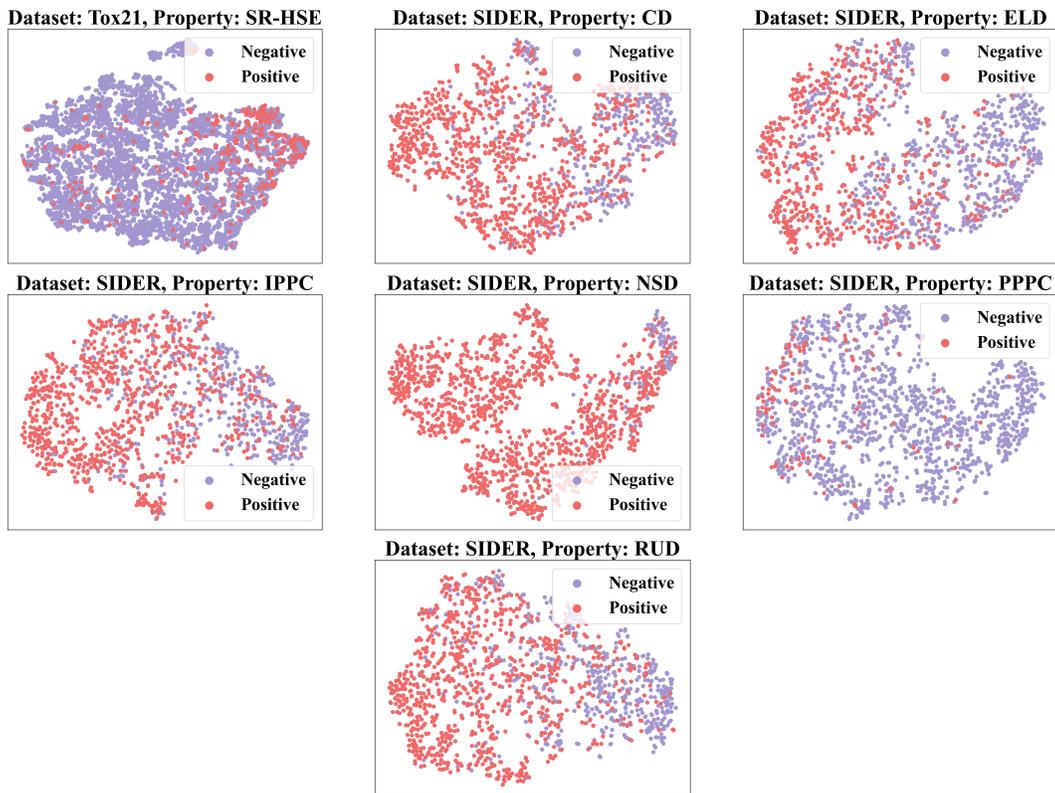
Figure 8: More molecular representations encoded by GS-Meta [73].
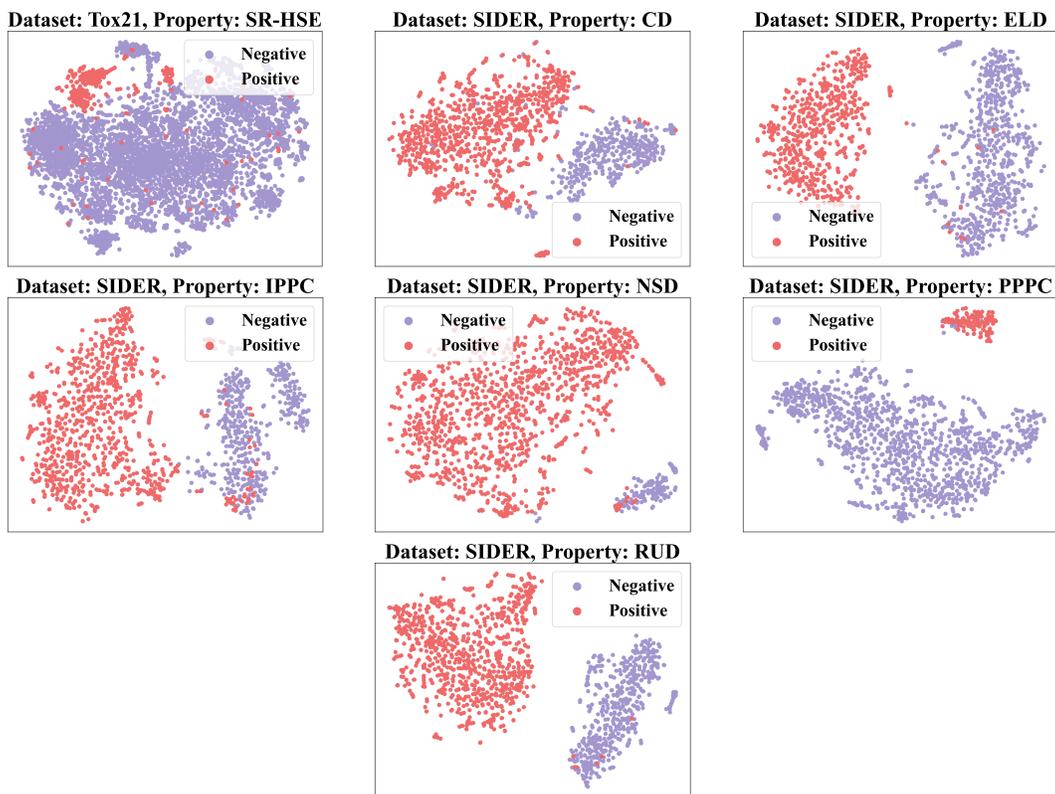
Figure 9: More molecular representations encoded by Pin-Tuning.

H   Limitations and future directions

As we discusses in Section 5.2, although our method significantly outperforms the state-of-the-art baseline method, our method exhibits higher standard deviations in the experimental results under multiple runs with different seeds.

We further speculate that these high standard deviations might be due to the uncertainty in the context information within episodes. The explicitly introduced molecular context, on one hand, provides effective guidance for tuning pre-trained molecular encoders, but on the other hand, this information also carries a high degree of uncertainty. We aim to model the target property through the molecule-property relationships within episodes, but each episode is obtained by sampling very few samples from the large space corresponding to the target property. The uncertainty between different episodes is relatively high. How to quantify and calibrate this uncertainty is another question worth exploring, which we will investigate in our future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We present our claims and main contributions in the abstract and introduction. We propose a parameter-efficient in-context tuning method, named Pin-tuning, to effectively adapt pre-trained molecular encoders to few-shot molecular property prediction tasks.

   Guidelines:
   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in Appendix H.

   Guidelines:
   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the derivation of our proposed Emb-BWC regularization, with related assumptions in Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The proposed method and experimental setups are clearly described. Additionally, the link of code and other details for reproducing are provided in Appendix F. The datasets are publicly available.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

Justification: The link of code is provided in the abstract, and other details for environment and configurations are provided in Appendix F. The datasets are publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to Section 5.1 and Appendix F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report error bars in our experimental results, such as in Table 1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Appendix F.1.

   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research conforms with the NeurIPS Code of Ethics.

   Guidelines:
   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our method addresses the few-shot molecular property prediction problem, which is important for drug discovery and material science, as discussed in the introduction and conclusion.

    Guidelines:
    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our research poses no such risks.

Guidelines:
- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly respect and credit for existing assets. We cite relevant papers and comply with the licenses.

Guidelines:
- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our code is well documented.

Guidelines:
- The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:
- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:
- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.