# Selecting Top-$k$ Data Science Models by Example Dataset

Anonymous Author(s)

## ABSTRACT

Data analytical pipelines routinely involve various domain-specific data science models. Such models require expensive manual or training effort and often incur expensive validation costs (*e.g.,* via scientific simulation analysis). Meanwhile, high-value models remain to be ad-hocly created, isolated, and underutilized for a broad community. Searching and accessing proper models for data analysis pipelines is desirable yet challenging for users who are not familiar with domain knowledge. This paper introduces **ModsNet**, a novel **MOD**el **S**electio**N** framework that only requires an **E**xample da**T**aset. (1) We investigate the following problem: Given a library of pre-trained models, a limited amount of historical observations of their performance, and an "example" dataset as a query, it aims to return $k$ models that are expected to have the best performance over the given dataset. (2) We formulate a regression problem and introduce a knowledge-enhanced framework using a model-data interaction graph. Unlike traditional methods, (1) ModsNet supports strict cold-start scenario (when a new dataset without any interaction with existing models is given). We introduce a dynamic, cost-bounded "probe-and-select" strategy to identify promising pre-trained models incrementally. (2) To reduce the learning cost, we develop a clustering-based sparsification strategy to prune unpromising models and their interactions. (3) We showcase of ModsNet built on top of a crowdsourced materials knowledge base platform. Our experiments over real-world analytical pipelines verified the effectiveness, efficiency, and applications of ModsNet.

## 1 INTRODUCTION

The emerging interests of data-driven analytical pipelines [22, 24] highlight the need to utilize various (pre-trained) *domain-specific* models effectively. Unlike large-scale models with abundant training data from Web, such models are typically learned from small-scale, domain task-specific experimental or simulation data (*e.g.,* material science, biology, chemistry, physics), requires expensive manual verification effort from trained domain experts, and take a great manual effort to tune due to high-dimensional in-situ measurements (*e.g.,* temperatures, processing methods, devices settings). In addition, validating the performance of domain-specific models can be costly due to the lack of "ground-truth", equipment access, and time-consuming simulation, which easily takes weeks or months.

One may want to fine-tune pre-trained domain-specific models [13] without training a new model from scratch. Nevertheless, high-quality, models remain to be ad-hocly created case by case, isolated in different sites, and remain unpublished and underutilized. Moreover, there lacks proper methods to make them "searchable", particularly for users who have little domain knowledge yet still need to ensure the performance of the models (*e.g.,* accuracy).

Some platforms, such as Hugging Face [15], Kaggle [2], and Github [1], gathered abundant pre-trained models, scripts, and associated datasets with search functions supported. For example, a research lab may publish its pre-trained models and related XRD datasets for peak-finding on Hugging Face. Nevertheless, whether a particular model performs well or can be adapted with fine-tuning for new XRD datasets from another lab remains unclear.

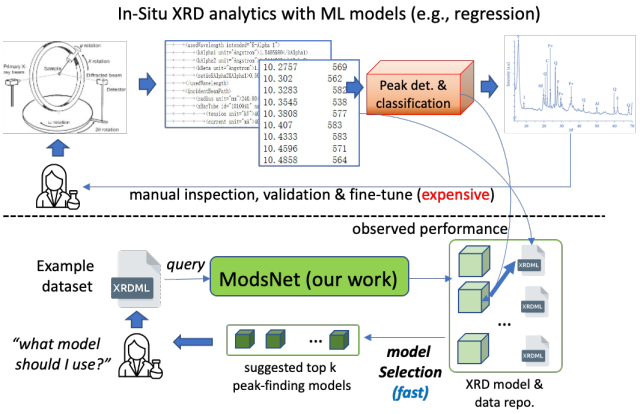


**Figure 1: Selecting "Peek-finding" models to support peak analysis in X-Ray Diffraction Data (XRD).**

**Example 1:** An material scientist has a new XRD sample dataset to be analyzed and wants to find proper models that can automate the process of peak detection, a cornerstone task that seeks for peaks to infer the crystalline structures. By accessing Hugging Face, she may issue a keyword query to find some pre-trained models for "Peak analysis". However, this alone limits the results to be content- or textual-relevant models, with no guarantee on *e.g.,* accuracy.

A more desirable case is a "*search by dataset*" feature: directly searching with the XRD dataset as a representative example. With a collection of pre-trained models and XRD datasets, as well as a set of historical observations of the models and their performance over the datasets, it is desirable to identify a set of top-k pre-trained peak-finding models, such that they are expected to have relatively high performance over the new dataset. Ideally, this search should *avoid* expensive ad-hoc inference tests and validation processes. Moreover, valuable auxiliary metadata should also be suggested (such as training setup, hyper-parameter settings, experimental parameters, and data providers) to help the querier fine-tune and validate the suggested models [22]. □

Such need is evident in promoting underutilized domain-specific models for accelerating scientific collaboration, and to improve the trustworthiness of the collaboration. *How to enable a "search by dataset" mechanism to find high-quality domain-specific models, such that they are expected to perform well for the "query" dataset?*

In particular, we are interested in the following problem:

- **Input:** a set of pre-trained models $\mathcal{M}$, a (limited) amount of historical performance $\mathcal{H}$, a model performance measure $P$, integer $k$, and an example dataset $d_q$ (a "query");
- **Output:** a set of $k$ pre-trained models from $\mathcal{M}$ that are expected to have good performance $P$ over $d_q$.

There are several possible approaches.

(1) Fine-tuning models for new data. Tools such as AutoML [8] provides automated tuning pipelines over feature and parameter space. However, tuning domain-specific models requires domain knowledge, and remains expensive to validate even with AutoML tools, leaving alone the large exploration space of parameters.

(2) Model selection has been studied to provide pre-trained models for fine-tuning in transfer learning [36]. These approaches often require actual re-training or inference tests of the pre-trained models. A majority of the work uses "transferability scores", a measure that only quantifies the compatibility and distribution similarity of different datasets. Instead, we aim to suggest pre-trained models with good user-defined performances, and avoid unnecessary testing that is inherently expensive for *e.g.,* scientific data analysis.

(3) Recommendation approaches can be applied to build a model that ranks and recommends models. A challenge remains to tackle the "cold-start" issue: how to recommend models for a dataset that has never been seen before? Robust recommendations should be made with limited historical observations.

In response, we propose **ModsNet**, a model selection framework to promote utilizing high-quality models. It is optimized with several unique features that are not addressed by existing methods.

*"Select by example dataset".* ModsNet only requires a sample dataset to be provided to suggest pre-trained models. It exploits data set features along with a set of "meta-features" including model sketches, training environment, testing data features and run-time performances to jointly predict the quality of pre-trained models without performing transfer learning, re-training or fine-tuning.

*"Cold-start" Selection.* ModsNet supports "cold-start" search for a new dataset that has not seen before at query time. Upon receiving a new example dataset, ModsNet incrementally explores relevant performance information, to decide a cost-bounded "probe" tests and make suggestions. This is in particularly feasible in practical search scenarios, where large amount of datasets ("query workload") requests a relatively smaller amount of models.

*Knowledge-preserving.* ModsNet is equipped with a knowledge graph with auxiliary model-data interaction environment. The knowledge graph is incrementally maintained to profile new models, datasets and test results. This unique architecture, featuring an evolving, knowledge-preserving repository, makes ModsNet sustainable as a self-evolving approach that preserves and "accumulates" new knowledge from each query. That is, it learns to select models better as more queries are issued in the long term.

**Related work**. We categorize related work as follows.

*GNN-Based Recommendation.* Graph neural networks have been specified to improve recommender systems [11, 33], which can be categorized into three types: User-item extension, such as NGCF[29], lightGCN [14], and INMO-GCN [34]; Social network enhanced [9, 31]; and Knowledge graph enhanced[26, 28].

GNN-based user-item extensions [14, 29] generally learn graph representation of a bipartite user-item graph in a transductive scenario to predict missing links, where all the nodes are assumed to have been seen. INMO-GCN [34] learns graph representation in an inductive manner without retraining. However, it still needs to require that any test node needs to be in the graph by default. Knowledge-enhanced methods [26, 28] aim to enrich the node features with *e.g.,* metapath features taken from an underlying knowledge graph. Similarly, social network enhanced methods enhance node features with side information from social networks. A common principle of these approach remains to solve a link prediction problem and select products with high probability.

ModsNet is equipped with a knowledge graph on factual knowledge and metadata for pre-trained models and datasets. It differs with existing methods with the following. (1) Rather than treating model search as a link prediction problem [26, 28] that does not directly address model quality, ModsNet solves a regression problem to explicitly quantify, estimate and optimize model performance, thus is more likely to select high-quality models, as verified in our experimental study. (2) The strict cold-start issue remains not discussed for these methods. ModsNet performs cost-bounded probe tests to strike a balance between search response time and model quality. (3) Direct learning from embedded bipartite structure of a mass of entities remain expensive. ModsNet uses a clustering-based sparsification strategy to reduce unpromising interactions. This allows fast response to large query workloads with small overhead.

*Model Selection for Transfer Learning.* Transfer learning, in simple terms is knowledge transferring between similar task domains[20], which is a practical way to use pre-trained models to learn new knowledge effectively and reduce the enormous training costs. Several methods have been proposed to estimate transferability based on the pre-trained model features and target dataset labels, such as NEC[25], LEEP[18], and LogME[36]. ModsNet can also be made use for this task with an advantage that it incurs cost-bounded inference tests for each candidate model over the target dataset.

## 2 MODEL SELECTION: A FORMULATION

We start with several notations used by ModsNet framework. We then provide a characterization of the model selection problem.

**Interaction Graph**. ModsNet works with a *model-data interaction graph* $\mathcal{G} = (\mathcal{M} \cup \mathcal{D}, \mathcal{I}, \mathcal{F})$, which is an attributed bipartite graph. (1) $\mathcal{M}$ is a set of model nodes, where each node $m \in \mathcal{M}$ refers to a user-registered data science model to be selected. A data science model can be a machine learning model, or a statistical model. (2) $\mathcal{D}$ is a set of dataset node, where each node $d \in \mathcal{D}$ refers to a dataset used for analytical tasks, *e.g.,* experimental data, simulation data, or measurement data. (3) $\mathcal{I} \subseteq \mathcal{M} \times \mathcal{D}$ refers to a set of *interaction* edges between models and dataset. Each edge $(m, d) \in \mathcal{I}$ denotes an observed test of a model $m \in \mathcal{M}$ over a dataset $d \in \mathcal{D}$.

*Attributes and features.* Each node $v \in \mathcal{M} \cup \mathcal{D}$ (resp. edge $(m, d) \in \mathcal{I}$) in $\mathcal{G}$ is assigned a tuple $\mathcal{F}(v)$ (resp. $\mathcal{F}(m, d)$) that encodes its attributes and values. (1) For a dataset node $v \in \mathcal{D}$, $\mathcal{F}(v)$ can carry data features such as measurements, parameter values, experimental data or simulation data; (2) For a model node $m \in \mathcal{M}$, $\mathcal{F}(v)$ may carry "meta-features" [21] such as statistics of model parameters, hyper parameters, training environment, test settings, etc. (3) Given an interaction $(m, d) \in \mathcal{I}$, $\mathcal{F}(m, d)$ may carry a weight that quantifies a performance measure $P$ of user's interests of a model $m \in \mathcal{M}$ over a dataset $d \in \mathcal{D}$, such as accuracy, $F_1$ score, or training cost.

The node and interaction features can further be enriched by high-order path or topological features [28]. We provide a list of examples ModsNet used in Section 6).

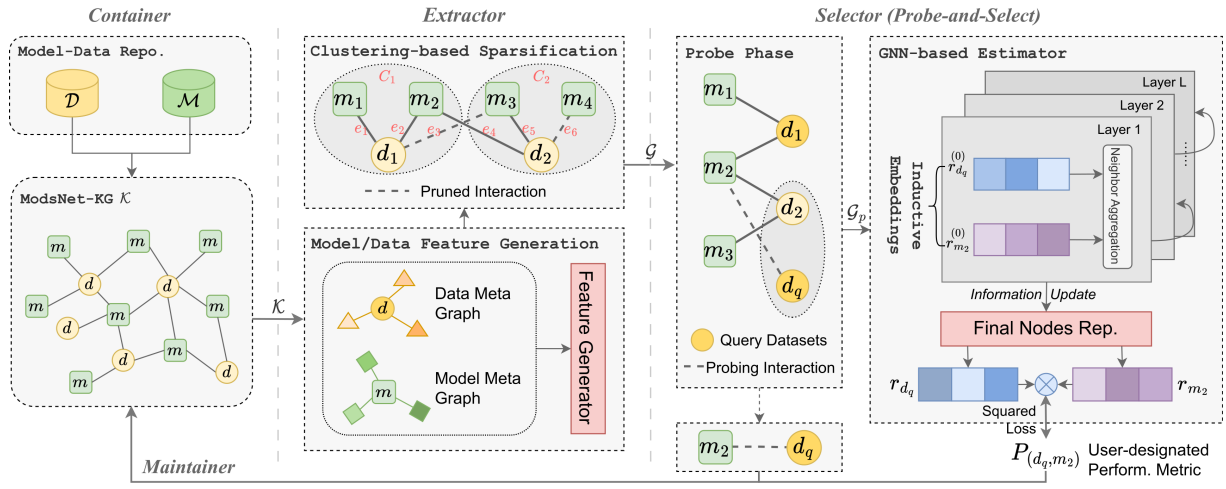


**Figure 2: ModsNet Framework and Architecture**

**Model-Data Knowledge Graph**. ModsNet is enhanced with a knowledge graph $\mathcal{K} = (\mathcal{V}, \mathbf{R})$ that uniformly encodes auxiliary factual knowledge relevant to data science models, datasets and their interactions. It consists of a set of real world entities $\mathcal{V}$ and their relations $\mathbf{R}$. Given an interaction graph $\mathcal{G}$, (1) each model node in $\mathcal{M}$ and dataset node in $\mathcal{D}$ from $\mathcal{G}$ has a counterpart entity in $\mathcal{V}$, (2) each $(m, d) \in \mathcal{I}$ has a counterpart relation in $\mathbf{R}$, and (3) in addition, $\mathcal{V}$ contains a set of relevant entities associated with data science models *e.g.*, contributors, libraries, providers, equipment, metadata, and datasets such as authors, source, references. These auxiliary entities in $\mathcal{K}$ are the main sources of the enriched attributes, features and side information for $\mathcal{M}$ and $\mathcal{D}$ nodes in $\mathcal{G}$ (see Section 7).

By default, ModsNet initializes a model-data interaction graph $\mathcal{G}$ simply as a node-induced subgraph of $\mathcal{K}$ with the nodes in $\mathcal{M} \cup \mathcal{D}$. We remark that ModsNet can cold-start with a $\mathcal{K}$ and $\mathcal{G}$ as simple as a single model node without other information, as will be discussed.

**"Search by Dataset"**. Given a knowledge graph $\mathcal{K}$ with a set of pre-trained models $\mathcal{M}$, datasets $\mathcal{D}$, and their interactions $\mathcal{I}$ that encodes tests and their results quantified by a performance measure $P$, an example dataset $d_q$ as a *query*, and integer $k$, we aim to select $k$ models from $\mathcal{M}$, such that they are likely to have good performance that is consistently measured by $P$ over $d_q$.

ModsNet characterizes the problem by solving a regression task. It aims to train a graph neural network (GNN)-based model (simply denoted as ModsNet) to minimize a model performance loss below:

$$\mathcal{L}(\mathcal{D}, \mathcal{M}, \mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{(d,m) \in \mathcal{I}} |\hat{P}_{(d,m)} - P_{d,m}|$$

Here $\hat{P}_{d,m} = f(\text{ModsNet}(d), \text{ModsNet}(m))$, where (1) ModsNet $(d)$ and ModsNet $(m)$ are the learned dataset and model representations, respectively, and (2) $f(\cdot)$ is a score function that transforms the representations to estimated model performance, consistently measured by a user-specified metric $P$ (*e.g.*, accuracy, learning cost). That is, it learns to minimize the differences between the observed performance $P_{d,m}$ and estimated counterpart $\hat{P}_{d,m}$, for each "model-data" interaction $(m, d) \in \mathcal{I}$. We present the details in Section 5.

Upon new dataset query $d_q$, ModsNet adapts to $d_q$ in an inductive manner without retraining, and predicts the performances of $\mathcal{M}$ over $d_q$. It chooses top the $k$ pre-trained models with the highest expected performances for downstream fine-tuning or inspection.

## 3 FRAMEWORK OVERVIEW

As illustrated in Fig. 2, ModsNet framework has four components: (1) a **Container** to manage a built-in model repository $\mathcal{M}$ and a dataset repository $\mathcal{D}$, (2) a model-data knowledge graph $\mathcal{K}$ maintained by the **Maintainer**, (3) an **Extractor** to construct the model-data interaction graph $\mathcal{G}$ from $\mathcal{K}$ with necessary feature generation; and (4) a **Selector** to select models from $\mathcal{M}$ for query datasets, aided by a GNN-based **Estimator** model.

The workflow of ModsNet goes through the major steps below.

(1) *One-time Initialization*. ModsNet cold-starts with a default $\mathcal{K}$, which can be as simple as a single pair of a model $m$ and a dataset $d$. The **Extractor** module then constructs an interaction graph $\mathcal{G}$.

(2) *Query-time Selection*. Upon receiving an example dataset $d_q$ (a query), **Selector** select models for $d_q$ with the following cases.

- (a) If $d_q \in \mathcal{D}$, and $\exists m \in \mathcal{M}$ such that $(d_q, m) \in \mathcal{I}$, *e.g.*, $d_1$, one of the query datasets in Fig. 2, linked with $m_1$ and $m_2$. ModsNet will skip the *Probe Phase* and invoke a typical transductive setting, readily predicting the performance to make regression-based ranking and selection with $\mathcal{G}$.
- (b) Otherwise, if $dq \in D$, but $\nexists m \in \mathcal{M}$ such that $(d_q, m) \in \mathcal{I}$, or if $d_q \notin \mathcal{D}$, which means $d_q$ is a new dataset without any existing interactions in $\mathcal{G}$, ModsNet will invoke the *Probe Phase* to perform a bounded number of "probing tests" to estimate the performances of models in $\mathcal{M}$ over $d_q$ inductively, by a *GNN-based Estimator*, with a slight delay cost.

(3) *Knowledge Integration*. Once processed, ModsNet invokes **Maintainer** to "recycle" $d_q$ and the results of probed tests, if any, as new interactions to enrich the knowledge graph $\mathcal{K}$ and interaction graph $\mathcal{G}$. This keeps $\mathcal{K}$ "fresh" and consistent with new attributes, features and and factual knowledge for future model selection.

We next introduce the Extraction module(**Extractor**) in Section 4, and the Selection module(**Selector**) in Section 5, respectively.

**Model:**
- **Metadata**: contributor, licenses, languages, task
- **Source code structure**: AST topological features
- **Training Record**: training dataset, base model, environment (CPU&GPU, o.s.), time cost, training performance
- **Model Info**: model type, # parameters, hyperparameters, layers' features, model size, flops, inference time per step(CPU/GPU), topological depth of the network

**Data:**
- **Metadata**: contributor, licenses, languages, organization, material sample, equipment, experiment settings: temperature, pressure, statistics of angles ($2\theta$), intensity ranges
- **Activity Record**: usability rating, hotness(# views, # votes, # downloads)
- **Statistics**: # classes, size categories
- **Description**: tasks/classes, textual descriptions

**Interaction:**
- **Model-data Pair**: model id, dataset id
- **Evaluation Record**: environment(GPU), testing cost
- **Metrics**: accuracy, balanced accuracy, AUC, f1_score, precision, recall, Cosine similarity, Jaccard similarity, hamming loss, log loss

**Figure 3: (Meta) Features used by** ModsNet

## 4 EXTRACTION MODULE

The Extraction module extracts and optimizes the bipartite Interaction graph $\mathcal{G}$. It mainly executes two tasks: (1) feature generation, to enrich node features with auxiliary knowledge from the knowledge graph $\mathcal{K}$; and (2) interaction sparsification, to prune unnecessary interactions and reduce the learning overhead of ModsNet.

### 4.1 Standadization and Feature Generation

ModsNet has built-in scripts to automate a data ingestion pipeline below: (1) accepts user-registered models, datasets and test reports, (2) uniformly document and store them into standadized JSON objects over cloud-based databases, (3) exploits a built-in ontology to extract node attributes, encode their values into (meta-)features, and adopts established indexing to optimize the access.

Fig 3 provides a non-exhaustive list of (meta-)features and attributes we have extracted from data science models, scripts and datasets from real-world data repositories (see Section 6). These attributes are transformed to their feature representation $\mathcal{F}$ via matching encoding techniques, such as integer encoding, Word2Vec[17], BoW for textual attributes, and ast2vec[19] for script structures. The enriched features are integrated into $\mathcal{K}$ and naturally carried over by the interaction graph $\mathcal{G}$ for downstream model selection.

### 4.2 Interaction Sparsification

ModsNet-based regression can be expensive when the interaction graph $\mathcal{G}$, if directly extracted from $\mathcal{K}$ remains large. On the other hand, interactions may be pruned due to their similarity, or low recorded performance. We consider three cases below.

- (1) multiple similar models that differ from some (meta) features and tested over a same dataset should yield similar representations [16], and (thus) similar good performance;
- (2) a model tested with multiple similar datasets contribute almost equally good performances;
- (3) a model already has good performance for some datasets recorded, and the rest interactions with low performance over other datasets are less useful.

Intuitively, Case (1) and (2) can be jointly captured by "quasi-bicliques" in the interaction graph $\mathcal{G}$, with dense interactions between a group of similar models $C_{\mathcal{M}}$ and a group of similar datasets $C_{\mathcal{D}}$. These interactions should be sparsified due to their similar contributions for model performance regression. Case (3) captures interactions with lower performance between a group $C_{\mathcal{M}}$ from Case (1) or Case (2) and another group $C'_{\mathcal{D}}$, given that $C_{\mathcal{M}}$ already have higher performance interactions over a matching group $C_{\mathcal{D}}$.

Based on this intuition, we introduce a two-level clustering-based interaction sparsification strategy to extract a small $\mathcal{G}$. (1) Given $\mathcal{K}$, we first directly extract an original interaction graph $\mathcal{G}_o$. We perform a node-level clustering to create similar groups of $\mathcal{M}$ and $\mathcal{D}$ nodes, and induces $\mathcal{G}_o$, where each group is treated as a single *group node*. (2)

formulate an problem of *model-data interaction sparsification problem*, denoted as MDIM.

Following this intuition, we introduce a two-level clustering-based interaction sparsification strategy to extract a small $\mathcal{G}$ in terms of the number of interactions. For this, we (1) cluster $\mathcal{M}$ and $\mathcal{D}$ to similar groups of models and datasets, respectively; and (2) create a second level of clustering of interactions that to prune "cross-clusters" interactions and sample those within each cluster,

*Symmetric Difference.* We adopt a node clustering strategy following a minimal disagreement principle [5]. In our context, the "disagreement" of clusters is quantified by a symmetric difference. Given a set of clusters that contains both model and dataset nodes $C = \{C_0, C_1 ... C_n\}$, and $\mathcal{G}$, the symmetric difference between $C$ and $\mathcal{G}$, denoted as $sd(C, \mathcal{G})$, is computed as

$$sd(C, \mathcal{G}) = \sum_{C \in C} \sum_{e \in \mathcal{M}_C \times \mathcal{D}_C} diff(e, C, \mathcal{G})$$

where $\mathcal{M}_C$ and $\mathcal{D}_C$ refer to the set of model node and dataset node in the cluster $C$, respectively. Here,

$$diff(e, C, \mathcal{G}) = \begin{cases} 1 & e \in \mathcal{G} \setminus (\mathcal{M}_C \times \mathcal{D}_C) \cup (M_C \times \mathcal{D}_C) \setminus \mathcal{G} \\ 0 & otherwise. \end{cases}$$

The symmetric difference evaluates the cost of "violating" the topology structure of the original bipartite $\mathcal{G}$. An intuition is that we pursue a clustering that favors interactions between models and datasets from the same cluster, as they connect a set of similar models and datasets; and penalize "cross-cluster" interactions, as these interactions may connect unrelated models and datasets.

**Problem Statement**. Given an attributed bipartite graph $\mathcal{G} = (\mathcal{M} \cup \mathcal{D}, \mathcal{I}, \mathcal{F})$, *model-data interaction minimization* aims to cluster $\mathcal{M} \cup \mathcal{D}$ to $C = \{C_0, C_1 ... C_n\}$ such that $SD(C, \mathcal{G})$ is minimized.

**Lemma 1:** *The* MDIM *problem is* NP*-hard.* □

The hardness of MDIM can be verified by a reduction from the 3-Exact-3-cover problem. This problem is known to be NP-complete [12]. (Anon.) We provide detailed proof in [10].

**Approximating Optimal Interaction Set**. We next present an approximation algorithm, denoted as APXIM. APXIM exploits a

"clustering-with-pruning" strategy to cluster model and data nodes and construct a sparsified model-data interaction graph $\mathcal{G}$.

(1) *Clustering*. Given an interaction graph $\mathcal{G}$, the "clustering" step firstly clusters model nodes and dataset nodes in $\mathcal{G}$ to a set of clusters $\mathcal{C} = \{C_1, C_2 ... C_n\}$. For each cluster $C_i$ in $\mathcal{C}$, we assume that all nodes in $C_i$ form a bi-clique. "clustering" returns the set of clusters that approximates the optimal symmetric difference. Here, "clustering" solves a model-data interaction minimization problem.

(2) *Pruning*. Given the clusters $\mathcal{C}$ from the "clustering" step, we firstly assume each of the clusters $C$ form a bi-clique. "pruning" step selects edges $e$ in the bi-clique for constructing model-data interaction graph $\mathcal{G}$ if the edge $e$ is in the original interaction graph and the $e$ is not similar to any of the previously selected edges.

**Interaction Correlation**. To evaluate the similarity between two edges, we introduce a new metric for the model-data interaction similarity $Sim(\cdot)$ as follow, given two interactions $e = (m, d)$ and $e' = (m', d')$,

$$sim(e, e') = \alpha \cdot ksim(m, m') + (1 - \alpha) \cdot cos(d, d')$$

Here, ksim quantifies the similarity of $m$ and $m'$, while "cos" computes the cosine similarity between feature vectors of $d$ and $d'$. In our problem, we evaluate the similarity of two edges with the linear combination of model similarity and cosine similarity of datasets. To emphasize the high-quality interactions and limit information redundancy, $\forall e, e' \in C$, if $sim(e, e')$ is larger than a threshold $\theta$ and e has a higher performance than $e'$, then $e'$ will be pruned.

**Approximation Algorithm**. The algorithm APXIM, outlined in Fig. 4, induces the attributed bipartite $\mathcal{G}_0$ from knowledge graph $\mathcal{K}$ (line 1). (1) In the clustering step, a) APXIM initializes a single cluster $C_{curr}$ by randomly selecting a model node $m_s$ from $\mathcal{G}_0.\mathcal{M}$ and adding $m_s$ and its neighbors in $\mathcal{G}_0$ to $C_{curr}$. APXIM then updates model node set $M$ by removing $m_s$ from $M$ (line 4-6); b) it then verifies each model node $m$ in $M$ to see if adding $m$ to $C_{curr}$ can minimize the symmetric difference. For this case, APXIM estimates the impact of including node $m$ to the overall symmetric difference based on its neighbors. Then, it exploits a randomization selection strategy to decide whether to include $m$ to $C_{curr}$. (lines 7-19); To this end, the clustering step solves a *bipartite correlation clustering* problem [4]. This step approximates the optimal clusters (See Theorem 2). (2) In the procedure Prune, for each cluster $C$, we assume model and dataset nodes in $C$ form a bi-clique. For each edge $e$ in the bi-clique, APXIM adds $e$ to $\mathcal{G}$, if a) $e$ is in $\mathcal{G}_0$; b) the similarities between $e$ and previously selected edges are less or equal to $\theta$.

**Example 2:** As the Extractor part in Fig. 2, with the featured model-data bipartite graph $\mathcal{G}$, APXIM: (1) clusters model nodes $\{m_1, m_2, m_3, m_4\}$ and dataset nodes $\{d_1, d_2\}$ into clusters $C_1$ and $C_2$, by invoking "clustering" step of APXIM (line 2-20 in Fig 4), (2) prunes edges $e_3$, as $e_3$ and $e_4$ are two "cross-cluster" interactions bridging C1 and C2, and the performance of $e_3$ is lower than $e_4$. (3) prunes edge $e_6$, as $sim(e_5, e_6) = 0.6$, which is larger than our pre-defined threshold $\theta$ 0.5, and $e_5$ performs better than $e_6$. □

**Theorem 2:** *Algorithm* APXIM *computes a model-data bipartite graph $\mathcal{G}$ which approximates an optimally minimized structure with a ratio of* 4, *in time* $O(|\mathcal{I}| \cdot (|\mathcal{M}||\mathcal{D}|))$. □

**Proof sketch:** **(I)** To see the approximation ratio of APXIM, we first prove that APXIM ensures a 4-approximation for the MDIM. Here we construct a reduction from MDIM to a *bipartite correlation clustering problem* (BCC) [4]. Given a bipartite graph $G = (L, R, E)$, where $L$ and $R$ are the set of nodes on the left and right sides of the bipartite graph $G$, respectively. $E$ refers to the edges in $G$, $E = \{e = (l, r) \mid (l, r) \in L \times R\}$. BCC computes a set of clusters of nodes $Cl = \{Cl_0, Cl_1 ... Cl_n\}$ such that (1) $\bigcup_{Cl_i \in Cl} Cl_i = L \cup R$; (2) Assuming each cluster $Cl_i$ forms a bi-clique, and all bi-cliques form a bipartite graph $G_B = (L_B, R_B, E_B)$, the symmetric difference between $G$ and $G_B$ , $cost(G, G_B)$ is minimized. It is known that BCC ensures a 4-approximation ratio.

*Reduction* Given an attributed bipartite graph $\mathcal{G} = (\mathcal{M} \cup \mathcal{D}, \mathcal{I}, \mathcal{F})$, we construct an instance of BCC $G = (L, R, E)$ as follows: $L = \mathcal{M}$, $R = \mathcal{D}$ and $E = \mathcal{I}$, and $cost(G, G_B) = SD(\mathcal{C}, \mathcal{G})$.

Given a solution $Cl$ of the above instance $(L, R, E)$ of BCC, we construct a solution $\mathcal{C}$ for MDIM, by setting $\mathcal{C} = Cl$. Let $Cl^*$ be the set of clusters with optimal symmetric difference $cost(Cl^*)$. It is known that $cost(Cl) \leq 4 \cdot cost(Cl^*)$ [4]. Between MDIM and BCC problem, we can observe that $SD(\mathcal{C}, \mathcal{G})$ can be directly mapped to $cost(Cl)$ by setting $E_B = \mathcal{M}_C \times \mathcal{D}_C$. It then suffices to show that $SD(\mathcal{C}, \mathcal{G}) \leq 4 \cdot SD(\mathcal{C}^*, \mathcal{G})$, where $\mathcal{C}^*$ is the set of clusters that has the optimal $SD$ objective value.

**(II)** For the time cost, APXIM takes at most $|\mathcal{I}|$ rounds. Each round verifies all available models to decide how to construct each cluster (line 7-19). Additionally, APXIM takes at most $|\mathcal{M}||\mathcal{D}|$ time to examine prune similar interaction edges (See Procedure Prune). Putting all together, APXIM takes $O(|\mathcal{I}| \cdot (|\mathcal{M}||\mathcal{D}|))$ time to construct model-data interaction bipartite graph, where $\mathcal{I}$ is the number of edges in induced graph $\mathcal{G}_0$ and $|\mathcal{M}|$ and $|\mathcal{D}|$ refer to the number of model nodes and data nodes in $\mathcal{G}_0$, respectively. □

This verifies theorem 2. (Anon.)We provide more details in [10].

## 5 SELECTION MODULE

The Selection module will select top-k pre-trained models for a query dataset $d_q$ based on the bipartite graph $\mathcal{G}$ refined by **Extractor**. The kernel of this module is the *Probe-and-Select Strategy*, with a GNN-based **Estimator** as the foundation. It is competent in strict cold-start scenarios (case(2)(b) in Section 3), and furnishes new factual knowledge for **Maintainer** into ModsNet-KG $\mathcal{K}$.

Next, we will break down the Probe-and-Select Strategy in section 5.1 and introduce GNN-based Estimator in detail in Section 5.2.

### 5.1 Probe and Select Strategy

As the name implies, the Probe-and-Select Strategy includes two parts: (1) Probe: for a new query dataset $d_q$, connect it with a set of models $\mathcal{M}_{prob}$ in $\mathcal{M}$ to create some probing interactions $\mathcal{I}_{prob}$. (2) Select: for each $m$ in $\mathcal{M}$, predict its performance on $d_q$ by a GNN-based **Estimator**. Given this, return models with top-$k$ predictions and dynamically maintain ModsNet-KG $\mathcal{K}$.

It has been observed in representational similarity metrics [16] that a model tends to produce similar representations over two datasets $d$ and $d'$, if $d$ and $d'$ are similar in terms of their representations. Following this, we designed *Probe-and-Select Strategy*. Fig. 5 shows the details of *Initialization* and *Probe Phase*.

**Algorithm** APXIM

*Input:* knowledge graph graph $\mathcal{K}$, threshold $\theta$.
*Output:* attributed bipartite graph $\mathcal{G}'$.

attributed bipartite graph $\mathcal{G}_0$ := Induce($\mathcal{K}$)
/* *induce model-data interaction graph from knowledge graph* $\mathcal{K}$ */
set $C := \emptyset$; set $M := \mathcal{G}_0.\mathcal{M}$;
**while** $M \neq \emptyset$ **do**
  model $m_s$ := randomSelect($M$);
  set $C_{curr} := m_s \cup$ neighbor($m_s$);
  $M = M \setminus m_s$;
  **for each** $m \in M$ **do**
    set $S_1$ := neighbor($m_s$) \ neighbor($m$);
    set $S_2$ := neighbor($m$) \ neighbor($m_s$);
    set $S_{1,2}$ := neighbor($m$) ∩ neighbor($m_s$);
    double $seed$ := random(0, 1);
    **if** $seed \in [0, min(\frac{|S_{1,2}|}{|S_{2,1}|}, 1]$;
      **if** $|S_{1,2}| \geq |S_1|$;
        $C_{curr} := C_{curr} \cup m$;
      **else if** $|S_{1,2}| < |S_1|$
        $C_{single} := \{m\}$;
        $C := C \cup C_{single}$;
      $M := M \setminus m$;
  $C := C \cup C_{curr}$;
  $\mathcal{G} :=$ Prune($C, \theta, \mathcal{G}_0$);
  **return** $\mathcal{G}$;

**Procedure** Prune($C, \theta, \mathcal{G}_0$,)

attributed bipartite graph $\mathcal{G}$ := empty graph;
**for** $c \in C$ **do**
  **for** $e \in \mathcal{M}_C \times \mathcal{D}_C$ **do**
    **if** $e \in \mathcal{G}_0$ **and** $\nexists\, e' \in \mathcal{G}, \text{Sim}(e, e') > \theta$ **do**
/* *retrieve and prune similar interactions* */
      $\mathcal{G} := \mathcal{G} \cup e$;
**return** $\mathcal{G}$;

**Figure 4: Algorithm** APXIM

*Initialization*. We construct a graph $\mathcal{G}_s = (\mathcal{M}_s \cup \mathcal{D}_s, \mathcal{I}_s, \mathcal{F}_s), \mathcal{G}_s \subseteq \mathcal{G}$, where $\mathcal{D}_s$ denotes a set of datasets similar to $d_q$, $\mathcal{M}_s$ denotes the candidate models for probing. Based on the similarities(e.g., Euclidean Distance) between $d_q$ and the datasets in $\mathcal{D}$, we pick $l$ datasets most similar to $d_q$ as $\mathcal{D}_s$. Then, induce a set of models connected to the datasets in $\mathcal{D}_s$ as $\mathcal{M}_s$, to serve as candidate models for probing. Also, retrieve the interactions $\mathcal{I}_s$, features $\mathcal{F}_s$ and historical performance $\mathcal{H}_s$ associated with $\mathcal{M}_s$ and $\mathcal{D}_s$ from $\mathcal{G}$.

*Probe Phase*. This phase incorporates the new datasets to cope with the cold-start problem. Given $\mathcal{G}_s$, $\mathcal{H}_s$ and $d_q$'s feature $\mathcal{F}(d_q)$, we will measure the quality of every potential probing interaction $(d_q, m)$ by scoring each candidate model $m$ in $\mathcal{M}_s$ as:

$$Score(m) = \frac{1}{|\mathcal{I}_s|} \sum_{i \in \mathcal{I}_s} sim((d_q, m), i) \times P(i)$$

That is, we calculate the Interaction Correlation (defined in Section 4.2) between $(d_q, m)$ and each interaction $i$ in $\mathcal{I}_s$, times the corresponding observed performance $P(i)$, and aggregate the results related to $(d_q, m)$ by taking the average to get $Score(m)$.

After that, we select the models with the highest top-$b$ scores from $\mathcal{M}_s$ as the probing models set $\mathcal{M}_{prob}$, then connect $d_q$ with each model in $\mathcal{M}_{prob}$ to create a set of probing interactions $\mathcal{I}_{prob}$.

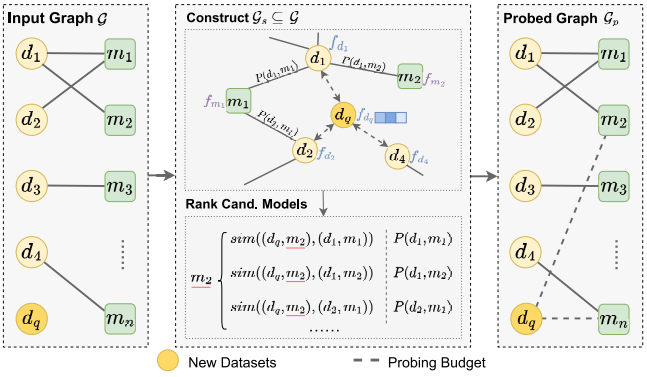


**Figure 5: Probe Phase**

At this point, we get the probed graph $\mathcal{G}_p = (\mathcal{M} \cup \mathcal{D}_p, \mathcal{I}_p, \mathcal{F}_p)$, where $\mathcal{D}_p = \mathcal{D} \cup \{d_p\}$, $\mathcal{I}_p = \mathcal{I} \cup \mathcal{I}_{probe}$, and $\mathcal{F}_p = \mathcal{F} \cup \{\mathcal{F}(d_q)\}$.

*Select Phase*. In this phase, we feed the probed graph $\mathcal{G}_p$ into a GNN-based **Estimator** trained on $\mathcal{G}$ for inference, to get the predicted performances of models in $\mathcal{M}$ over $d_p$. Then return the top-$k$ models that perform best on $d_q$. Also, the predicted results with the corresponding interactions we get at this step will be collected by **Maintainer** to enrich the ModsNet-KG $\mathcal{K}$ dynamically.

**Time cost**. We use quick-sort for sorting and store graph in an adjacency list. So that, in *Initialization*, the time cost includes $O(|\mathcal{D}|)$ for similarities calculations, $O(|\mathcal{D}| \log |\mathcal{D}|)$ for getting $\mathcal{D}_s$ and $O(|\mathcal{D}_s| + |\mathcal{I}_s|)$ for $\mathcal{G}_s$ construction. In *Probe*, it is $O(|\mathcal{M}_s|)$ for scoring models and $O(|\mathcal{M}_s| \log |\mathcal{M}_s|)$ for selecting probing models. In *Select*, it is $O(|\mathcal{M}|)$ for inference and $O(|\mathcal{M}| \log |\mathcal{M}|)$ for selecting top-$k$ models. In practical, there are always many more datasets than models, so the overall time cost would be $O(|\mathcal{D}| \log |\mathcal{D}|)$.

## 5.2 GNN-based Estimator

The foundation of our Selection module is a *GNN-based Regression Model*, which serves as an **Estimator** to estimate the performance of a particular model $m$ over a given query dataset $d_p$, even $d_p$ is unseen in the training graph.

**Inductive Embedding Generation layer**. In order to generate the global inductive embeddings for all old and new datasets involved in the training or testing part incrementally, we design a pre-module at the top of our GNN architecture, as an *Inductive Embedding Generation layer*. This layer reform the Inductive Embeddings Generation module of [34], which requires at least some observed interactions at the test phase to cope with new users; ours is equipped to generate the global embeddings for new dataset $d_p$ inductively without any real interactions by seamless fusion with the Probe strategy, which is described in Section 5.1.

Upon the graph $\mathcal{G}_p = (\mathcal{M} \cup \mathcal{D}_p, \mathcal{I}_p, \mathcal{F}_p)$ produced by the *Probe Phase*, for any $m \in \mathcal{M}$ and $d_q \in \mathcal{D}_p$, we select several template models and datasets which have similar interaction patterns to $m$ and $d_q$ from $\mathcal{M}$ and $\mathcal{D}$, and aggregate their embeddings by a weighted sum approach to get $e_{m_t}$ and $e_{d_t}$. Then generate the

inductive embeddings $e_m$ and $e_{d_q}$ for $m$ and $d_q$ as:

$$e_m = \frac{1}{(|\mathcal{D}_m|+1)^\alpha}\Big(\sum_{d\in\mathcal{D}_m} e_d + e_{m_t}\Big),$$
$$e_{d_q} = \frac{1}{(|\mathcal{M}_{d_q}|+1)^\alpha}\Big(\sum_{m\in\mathcal{M}_{d_q}} e_m + e_{d_t}\Big), \quad (1)$$

where $\mathcal{D}_m$ denotes a set of datasets interacted with $m$, $\mathcal{M}_{d_q}$ denotes a set of models interacted with $d_q$. $\alpha$ is used to dynamically control the degree of normalization. In the training phase, $\alpha$ gradually increases from 0.5 to 1. In the test phase, $\alpha$ is fixed as 1.

**Graph Convolution Layers**. ModsNet adopts graph convolution layers following a typical construction of GNN-based recommendation methods [6, 14, 29, 32]. As observed in prior work, (1) trainable weight matrix in the common designs of graph convolutional layer imposes a negative effect for recommendation purposes; and (2) nonlinear activation has no positive effect on recommendation task. ModsNet uses a linear weighted sum aggregator for two main reasons. Firstly, this aggregator allows for fast computation of information propagation, making it suitable for *Query-time Selection*. Secondly, it helps to mitigate the problem of over-smoothing when capturing high-order relations in the graph [14].

For the performance prediction loss, we employ the mean squared loss as the objective function. It encourages ModsNet to predict performance scores with the observed performance measurements as closely as possible. Then, in the testing phase, it sorts the model nodes according to the predicted performance score to recommend the top $k$ models with predicted best performances. The loss function formulates as follows:

$$\mathcal{L}_{\text{pred}} = \frac{1}{|\mathcal{I}_{\text{train}}|}\sum_{(d_i,m_j)\in\mathcal{I}_{\text{train}}} (\hat{P}_{(d_i,m_j)} - P_{(d_i,m_j)})^2 + \lambda\|\Theta\|_2^2 \quad (2)$$

$\mathcal{L}_{\text{pred}}$ further adds a $L_2$ regularization term parameterized by $\lambda$ to prevent overfitting. $\Theta$ denotes all trainable model parameters.

**Cost analysis**. The only model parameters are embeddings for all nodes on the 0-th layer. This makes ModsNet feasible to be trained for large graphs. The space complexity is $O((s+n)d)$, where $s = |\mathcal{D}|+1$ and $n = |\mathcal{M}|$ denote the number of datasets and models, respectively, $d$ is the dimension of the trainable node embeddings.

## 6 PROTOTYPE SYSTEM

As a proof of concept, we have built a prototype system of ModsNet. The system is supported by our established CRUX knowledge graph system [27]. CRUX is a crowdsourced scientific knowledge graph system that integrates SciML entities for material sciences. The prototype ModsNet is supported by CRUX in the following.

(1) *Knowledge Graph.* The knowledge graph $\mathcal{K}$ encodes the following three types of SciML entities: factual triples from materials data science (*e.g.,* elements, atoms, components), SciML models for peak finding (denoted as PKZoo), and related crowdsourced XRD datasets, along with additional entities (*e.g.,* experimentalist, universities, organizations, companies). A fraction of $\mathcal{K}$ that encodes PKZoo models and datasets are illustrated in Figure 6, which embeds a model-data bipartite interaction graph that contains an edge from a model node "pf_scipy_prom200" to an XRD 2D dataset "10Bi(Cd0.5Ti0.5).xrdml" with recorded accuracy on peak finding.

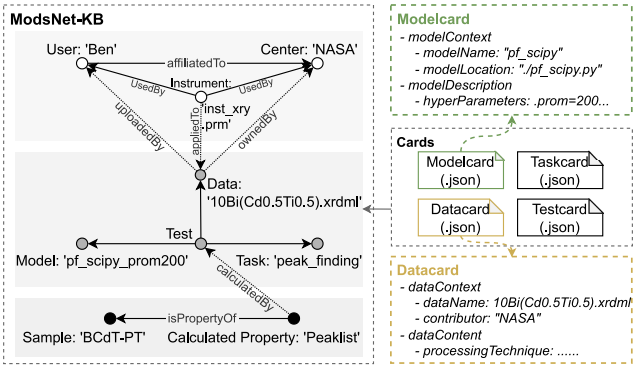


**Figure 6: A fraction of ModsNet-KB**

To evaluate ModsNet, we have enriched $\mathcal{K}$ with SciML entities from two more data sources: a set of image classifiers and images from Kaggle (KIZoo), and a collection of textual classification models from Hugging Face (HFZoo). The knowledge graph serves the role as the model-data knowledge graph $\mathcal{K}$ for ModsNet.

(2)*Attributes and Features.* The system is implemented with the following categories of attributes and features.

(3) *Storage Layer.* The system uniformly stores SciML entities in $\mathcal{K}$ as JSON objects with a standardized format called "cards". We maintain mainly 5 types of cards objects: model cards, data cards, user cards, test cards (for interactions), task cards (for task description), all managed in MongoDB. The raw datasets and model registration are supported by Web interface on top of Google Cloud.

(4) *Query Interface.* A user-friendly Web interface is developed to support the model search by example SciML dataset. Users can either specify or upload example dataset and requests promising pretrained SciML models. The registration and integration of datasets into $\mathcal{K}$ (**Maintainer**) is supported by an internal information extraction toolkit CRUX-IE [27] of CRUX.

The prototype system is deployed as part of the CRUX project. Currently, $\mathcal{K}$ has 2121 models, 18 datasets, 31826 interactions, and in total 65,850 SciML entities and 167,694 relations.

## 7 EXPERIMENT STUDY

### 7.1 Experiment settings

We experimentally evaluate ModsNet with real-world datasets. We investigate the following three questions.

- (**RQ1**): The effectiveness of ModsNet in estimating the performance of SciML models over query datasets, and regression-based model selection;
- (**RQ2**): How well ModsNet performs with the varied amount of observed performances, the quality of SciML models, and the amount of allowed "probes"?
- (**RQ3**): The efficiency of ModsNet, in terms of the training cost, and response time to new datasets as query workload.

In addition, we perform case analysis to verify (1) the effectiveness of clustering-based model-data bipartite graph extraction, and (2) application of ModsNet in real applications for peak finding and image classification.

| Dataset | # models | # datasets | # interactions | # (meta) features | density | task | performance metric | # probes | k | $\delta$ | $\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| KIZoo | 1800 | 72 | 9304 | 41 | 0.07179 | image classification | balanced accuracy | 50 | 10 | 0.8 | 100% |
| PKZoo | 462 | 289 | 98257 | 21 | 0.73591 | peak finding | F1_score | 200 | 10 | 0.8 | 100% |
| HFZoo | 932 | 66 | 974 | 13 | 0.01583 | text classification | accuracy | 10 | 10 | 0.8 | 100% |

**Table 1: Datasets Information**

**SciML models, Datasets and Knowledge graphs**. We have collected the following three real-world SciML model-data repository, and constructed their knowledge graphs with available metadata.

(1) KIZoo is a collection of Image Classification models along with their training and testing images from Kaggle [2].

(2) PKZoo consists of a repository of peak finding models, and a set of XRD datasets. The models and dataset are crowdsourced from CRUX community of 6 collaborative institutions. Our CRUX community also includes International Centre for Diffraction Data (ICDD) and JADE database user group (with active users from 53 countries) that provide ground truth for XRD peaks.

(3) HFZoo consists of a repository of text classifiers and text datasets from Hugging Face [15].

The details of the datasets are summarized in Table 1. All the SciML models, datasets, relevant SciML entities and relationships have been uniformly stored as JSON objects as "Cards" objects and integrated into a single ModsNet knowledge graph $\mathcal{K}$ (see Section 6).

**Model Selection Methods**. We have implemented 9 methods and adapted them to model selection, categorized as follows.

(1) ModsNet, supported by the prototype system; and ModsNet-C, its counterpart from a sparsified model-data interaction bipartite graph via model-data interaction pruning.

(2) Three state-of-the-art GNN-based methods: lightGCN [14], a light-weighted graph convolutional network (GCN) model that exploits linear propagation and uses weighted sum function to learn and update node embeddings for collaborative filtering; IDCF-GCN [32], an inductive GCN learning framework that integrates matrix factorization and attention-based structure learning to predict links and their weights; INMO-GCN [34], an inductive GCN model that considers interactions between preselected template items to learn user-item embedding, to mitigate the impact of *e.g.,* neighborhood bias.

These approaches extend the user-item bipartite graph-based recommendation to GNN-based link prediction, and recommend models with links having the highest probabilities. As none of them is adaptive to "cold-start" and require test nodes to be already a part of the model-data bipartite graph, we favored them by equipping all with the same "probe-and-select" module as in ModsNet. Meanwhile, all GNN-based baselines cannot simply treat an interaction with a low-performance score $P$ as a valid link. We add a simple threshold-based pruning strategy as a pre-processing step.

(3) Two non-GNN based collaborative filtering methods: CF [3], which utilizes integration information to predict the rate matrix and then make predictions based on the similarity between the datasets; and Matchbox [23], which combined collaborative filtering and content-based filtering, solved the cold-start problem by leveraging the side information (features) of the nodes. As the performance of CF varies significantly over different datasets, we favored CF by allowing it to *always* "warm-start" with a one-time cost to create a user-item matrix with no empty entries (or a fully connected bipartite graph) for model selection.

(4) LinearRegression, a simple linear regression model, and Wide&Deep [7], a supervised model that generalizes linear models with a jointly learned deep neural networks to combine memorization and generalization for recommendation. It is a flexible model that can be used for both regression and classification tasks, depending on the activation function on the output layer and the loss function of the Deep part. Here we implemented it as a regression method to predict the estimated accuracy and rank the model list based on that. Unlike GNN-based methods, both LinearRegression and Wide&Deep approach model selection by regression over observed model performance.

**Evaluation metrics and factors**. We evaluate ModsNet and baseline methods in effectiveness and efficiency.

For effectiveness, we report accuracy@k and NDCG@k given the ground truth top-k models from a *relevant set* of pre-trained SciML models $\mathcal{M}$ over the query datasets. Here we configure the relevant set by controlling a performance lowerbound $\delta$. For example, $\delta = 0.7$ means the selection is constrained to select pre-trained SciML models from a relevant set with validated performance (*e.g.,* balanced accuracy; see Table 1) at least 0.7 over any query dataset. This ensures that we fairly evaluate all the methods in identifying and suggesting "high-quality" SciML models.

We also control the amount of interactions of model-data bipartite graphs with a factor $\theta$, the ratio to the initial amount of interactions in the original knowledge graph $\mathcal{K}$ between all the models $\mathcal{M}$ and datasets $\mathcal{D}$. We use random sampling to tune the size. The default settings of $k$, $\theta$ and $\delta$ are reported in Table 1.

For efficiency, we report the training time of ModsNet, and the response time for a given set of queries.

**Environment**. All Experiments were executed on a Unix environment with Intel 2.6GHz CPUs, and 16GB memory. Each experiment was run 5 times with random query datasets and the mean results were reported.

## 7.2 Experiment results

**Exp-1: Accuracy@k (RQ1)**. We report the accuracy@k and NDCG@k of all the methods over KIZoo in Table 3 and PKZoo in Table 2. We set the performance threshold 0.8 for relevant sets. Table 3 over KIZoo tells us the following.

(1) In all the cases, ModsNet and ModsNet-C outperforms all other approaches in model selection.

(2) ModsNet-C achieved a comparable performance with ModsNet,

| metrics | Precision@5 | Precision@10 | Recall@5 | Recall@10 | NDCG@5 | NDCG@10 |
|---|---|---|---|---|---|---|
| ModsNet | **0.938** | **0.874** | **0.118** | **0.201** | **0.950** | **0.906** |
| ModsNet-RProb | 0.867 | 0.733 | 0.112 | 0.186 | 0.859 | 0.777 |
| ModsNet-C | TBF | TBF | TBF | TBF | TBF | TBF |
| ModsNet-NoKG | 0.313 | 0.507 | 0.070 | 0.147 | 0.310 | 0.452 |
| CF | 0.882 | 0.797 | 0.092 | 0.168 | 0.875 | 0.815 |
| Wide&Deep | 0.79 | 0.687 | 0.084 | 0.14 | 0.808 | 0.726 |
| lightGCN | 0.759 | 0.654 | 0.07 | 0.122 | 0.749 | 0.674 |
| Matchbox | 0.641 | 0.61 | 0.093 | 0.162 | 0.701 | 0.655 |
| IDCF-GCN | 0.677 | 0.574 | 0.077 | 0.135 | 0.679 | 0.607 |
| INMO-GCN | 0.615 | 0.549 | 0.068 | 0.11 | 0.592 | 0.549 |
| LinearRegression | 0.528 | 0.482 | 0.033 | 0.058 | 0.484 | 0.465 |

**Table 2: Effectiveness:** PKZoo

| metrics | Precision@5 | Precision@10 | Recall@5 | Recall@10 | NDCG@5 | NDCG@10 |
|---|---|---|---|---|---|---|
| ModsNet | **0.78** | **0.78** | **0.198** | **0.366** | **0.724** | **0.747** |
| ModsNet-RProb | 0.74 | 0.75 | 0.175 | 0.332 | 0.707 | 0.723 |
| ModsNet-C | 0.7 | 0.7 | 0.154 | 0.306 | 0.668 | 0.677 |
| ModsNet-NoKG | 0.72 | 0.660 | 0.186 | 0.322 | 0.692 | 0.665 |
| Matchbox | 0.64 | 0.68 | 0.149 | 0.3 | 0.602 | 0.642 |
| Wide&Deep | 0.6 | 0.59 | 0.147 | 0.269 | 0.591 | 0.589 |
| INMO-GCN | 0.52 | 0.59 | 0.145 | 0.278 | 0.514 | 0.564 |
| CF | 0.52 | 0.54 | 0.147 | 0.293 | 0.513 | 0.533 |
| IDCF-GCN | 0.48 | 0.48 | 0.123 | 0.225 | 0.483 | 0.479 |
| lightGCN | 0.48 | 0.45 | 0.093 | 0.178 | 0.46 | 0.444 |
| LinearRegression | 0.4 | 0.32 | 0.098 | 0.141 | 0.357 | 0.321 |

**Table 3: Effectiveness:** KIZoo

yet with a sparsified model-data interaction graph with 31% interactions pruned, and reduced the learning cost ofModsNet by 22.85%. This verifies the effectiveness of the optimization.

(3) Matchbox, Wide&Deep and CF perform better than the rest GNN-based link recommendation approaches INMO-GCN, lightGCN and IDCF-GCN. While the latter all capture topological features and nonlinear representations, they make suggestions only with links having high probability, thus may be more sensitive to neighborhood bias. For example, they may miss the chance to hit high quality models yet with sparse neighbors (fewer observations) or introduce lower quality models that are connected to some high quality counterparts.

(4) INMO-GCN works best among the three GNN-based link recommendation approaches. As verified in [34], INMO-GCN can outperform lightGCN with fewer parameters by only considering a subset of nodes from the interaction graph. ModsNet is optimized consistently to select the model and data nodes (see Section **??**).

The results in Table 2 are consistent. It is worth noting that while CF achieves good performance in particular over PKZoo, its performance is not stable for other datasets. We observed that CF benefited greatly from the dense initial interactions over PKZoo. As shown in Table 1, PKZoo is much denser than the other two datasets. On the other hand, it incurs much higher preprocessing cost, as will be discussed.

**Exp-2:Impact of factors (RQ2)**. We next investigate the impact of the performance lowerbound $\delta$, interaction ratio $\theta$ and $k$ to the effectiveness of ModsNet. We report the performances of the methods we introduced in Sec 7.1.

*Varying $\delta$*. Fixing $k$ = 10 and $\theta$ = 100%, we varied the threshold $\delta$ from 0.5 to 0.9 and reports the NDCG@10 in Fig. 7. For all the methods, it is harder to select SciML models with higher quality, as a higher threshold will make the relevant set smaller.

*Varying $k$*. As illustrated in Fig. 7, most of the methods perform better as $k$ varies from 3 to 18. As expected, selecting larger amount of pre-trained models allows ill-ranked models to contribute less

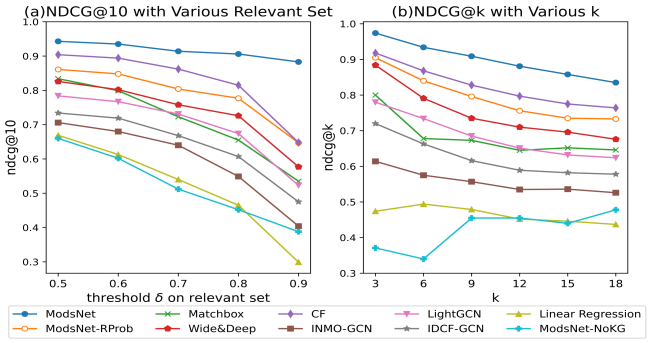


**Figure 7: Effectiveness: impact of relevant set and $k$ (PKZoo)**

in NDCG measures [30]. The performance of ModsNet is less sensitive to $k$, and remains the best among all the baseline methods. On the other hand, the performance of lightGCN degrades over larger $k$. A possible reason is that lightGCN trades efficiency with light-weighted features, and is more sensitive to the impact of non-important data and model nodes as $k$ is larger [34].

*Varying $\theta$*. We next evaluate the impact of the amount of available interactions. Fixing $k$ = 10 and $\delta$ = 0.8, we varied $\theta$ from 20% to 100% and report the results in Fig. **??**, over KIZoo and HFZoo. (1) Most methods perform better over larger amount of model-data interactions. Indeed, they in general benefit from more supervised information. For example, the supervised Wide&Deep and lightGCN are sensitive and greatly benefits from more enriched node and topological features from more interactions. (2) ModsNet illustrates robust performance over all cases, and outperforms other baselines in most cases as $\theta$ is larger.

*Varying training set quality*. We next investigate the impact of pre-trained models $\mathcal{M}$. Similarly as configuring relevant set with quality lowerbound $\delta$, we set a quality lowerbound for the training models $\mathcal{M}$ in the model-data bipartite graph $\mathcal{G}$ to train ModsNet. As GNN-based methods all require a high fixed threshold to get reasonable recommendation performance e.g. $\delta$=0.8. $\delta$ is not an impact factor for them, we omit their results. Fig. 8 verifies that ModsNet maintains a robust performance with good accuracy ($\geq$ 0.7) and can effectively learn from pre-trained models with larger variance of performances. In contrast, other approaches illustrate large variance in their performance over training sets with different quality. This indicates the applicability of ModsNet in practice, especially when it is often hard to control the quality of SciML models, or high-quality models are not always available.

*Varying probe budget*. In this test, we evaluate the effectiveness of the "probe-and-select" module over the number of interactions that are allowed to be performed. We only report the performance of GNN-based baselines for a fair comparison, as all four methods are uniformly enhanced by this strategy. We observe that they do not always benefit from larger budget and new probed interactions. This is because not all neighborhoods that are enforced to be added are important, and when certain amount of less important nodes are "forced" to be introduced to the scope, the accuracy of these methods start to degrade at some point.

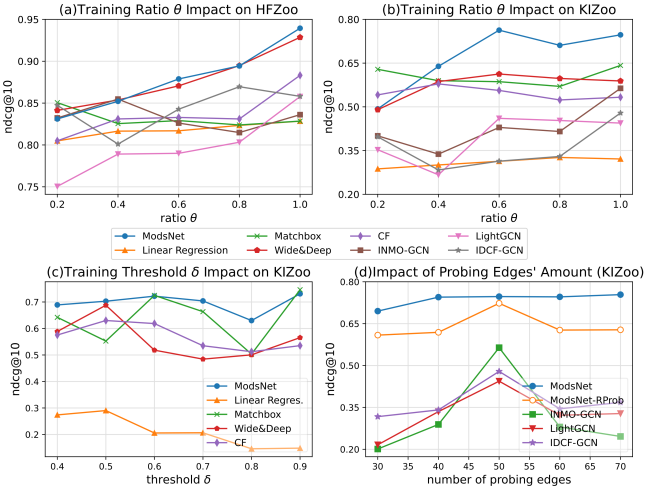


**Figure 8: Factors' Impact**

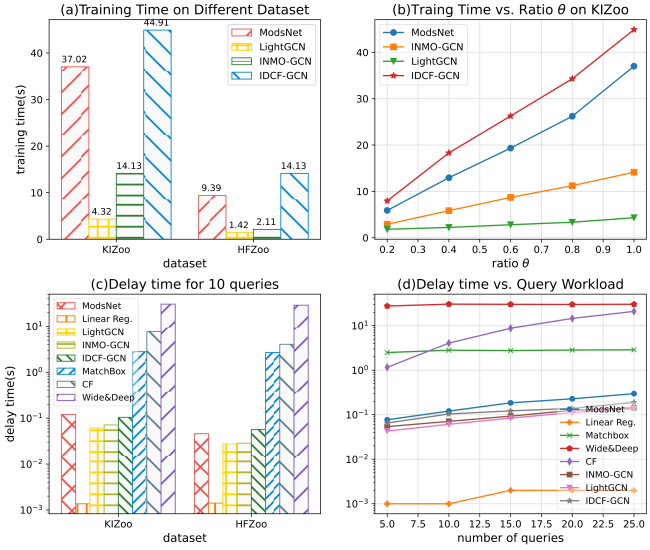


**Figure 9: Efficiency evaluations**

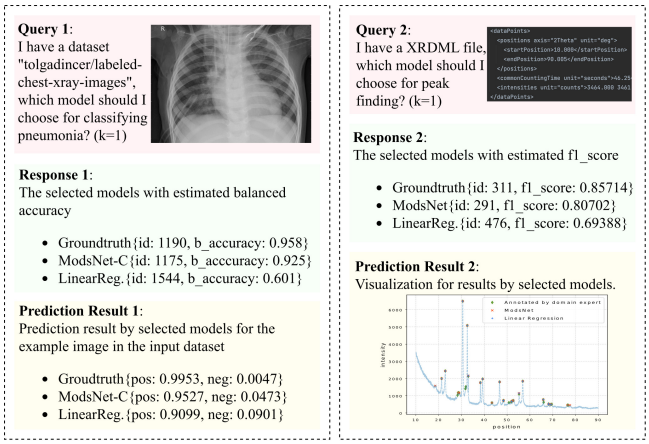


**Figure 10: Case study on application scenarios**

**Exp-3:Efficiency (RQ3)**. In this set of tests, we report the learning cost and query response time of ModsNet.

*Query response time.* We first compare ModsNet and all other methods in response to a query workload of 10 example datasets. Fig. **??** shows us the following. (1) ModsNet is comparable with GNN-based link recommendation models. In all cases, it takes 0.1 seconds to recommend top-10 models for 10 datasets. (2) For KIZoo, it outperforms Matchbox, CF and Wide&Deep by 23.25, 33.64 and 239.09 times. Indeed, to maintain a good recommendation accuracy for new datasets, CF always incurs an overhead on obtaining a fully evaluated Model-data matrix as in a user-item matrix. (3) While LinearRegression incurs the smallest cost, it does not provide desired selection accuracy in most of the cases as verified earlier.

*Varying the size of query workload.* Keep the default setting as in Table 1, we varied the query workload size from 5 to 25. As shown in Fig. **??**, ModsNet scales well with query workloads, and takes no more than 0.3 seconds to suggest all the answers. CF is the most sensitive methods and takes more time as query workload becomes larger. ModsNet improves CF better for larger query workload.

*Training cost.* Fig. **??** reports the impact of training cost of GNN-based methods. We found the following. (1) It is feasible to learn ModsNet. For KIZoo with 1800 models and 9304 interactions, it takes less than 40 seconds. As verified earlier, ModsNet-C further reduces the learning cost by 22.85% without sacrificing much in selection effectiveness. (2) All models take longer to train over more interactions, as expected. On the other hand, ModsNet provides the best results that outperforms other GNN-based counterparts, as verified earlier.

**Exp-4: Case Study**. We next perform case studies to evaluate ModsNet with real SciML applications, as illustrated in Fig. 10.

*Model selection for scientific image classification.* In the first case, a query (medical image of chest X-ray from Kaggle [35]) is issued to ModsNet, and it returns a set of $k$ pre-trained image classifiers from KIZoo with historically observed accuracy at least 0.8 over all registered datasets. The selected model correctly labeled the image. The classifiers suggested by other methods *e.g.*,LinearRegression has low performance on labeling the same image.

*Model selection for XRD peak finding.* A second case considers a query that issues an XRD data. We plot the 2D data in Fig. 10 with validated ground truth of real peaks annotated by domain scientists. ModsNet can recommend a pre-trained peak finding model that finds the peaks with $F\_1$ score at 0.807. On the other hand, the best model selected by LinearRegression has lower accuracy and misses major peaks.

## 8 CONCLUSION

We have investigated the problem of model selection with example dataset. We introduced ModsNet, a knowledge-enhanced framework that exploits graph learning over an interaction bipartite graph, with an auxiliary knowledge graph to select promising pre-trained models. We have also developed effective strategies to improve the learning efficiency with provable optimality guarantees, and to cope with cold-start scenarios. Our experimental

study verified the effectiveness and efficiency of ModsNet. These result verified its application for model selection in analytical applications. A future topic is to deploy ModsNet over distributed data, and extend it to more domain-specific applications.

## REFERENCES


[1] [n.d.]. Github. https://github.com/
[2] [n.d.]. Kaggle: Your Home for Data Science. https://www.kaggle.com/
[3] 2023. Collaborative filtering. https://en.wikipedia.org/w/index.php?title=Collaborative_filtering&oldid=1142286125
[4] Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke Van Zuylen. 2012. Improved approximation algorithms for bipartite correlation clustering. *SIAM J. Comput.* (2012).
[5] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine learning* (2004).
[6] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.
[8] Radwa Elshawi, Mohamed Maher, and Sherif Sakr. 2019. Automated machine learning: State-of-the-art and open challenges. *arXiv preprint arXiv:1906.02287* (2019).
[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.
[10] full version [n.d.]. ModsNet(Full Version). https://crux-project.github.io/ModsNet_Full.pdf
[11] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2022. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *ACM Transactions on Recommender Systems* (2022).
[12] Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*.
[13] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250.
[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.
[15] Hugging Face AI [n.d.]. Hugging Face – The AI Community Building the Future. https://huggingface.co/
[16] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International Conference on Machine Learning*. 3519–3529.
[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[18] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. 2020. Leep: A new measure to evaluate transferability of learned representations. In *International Conference on Machine Learning*. PMLR, 7294–7305.
[19] Benjamin Paassen, Jessica McBroom, Bryn Jeffries, Irena Koprinska, Kalina Yacef, et al. 2021. Mapping python programs to vectors using recursive neural encodings. *Journal of Educational Data Mining* 13, 3 (2021), 1–35.
[20] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.
[21] Adriano Rivolli, Luís PF Garcia, Carlos Soares, Joaquin Vanschoren, and André CPLF de Carvalho. 2022. Meta-features for meta-learning. *Knowledge-Based Systems* 240 (2022), 108101.
[22] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. 2020. Explainable machine learning for scientific insights and discoveries. *Ieee Access* 8 (2020), 42200–42216.
[23] David H Stern, Ralf Herbrich, and Thore Graepel. 2009. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*. 111–120.
[24] Jeyan Thiyagalingam, Mallikarjun Shankar, Geoffrey Fox, and Tony Hey. 2022. Scientific machine learning benchmarks. *Nature Reviews Physics* 4, 6 (2022), 413–420.
[25] Anh T Tran, Cuong V Nguyen, and Tal Hassner. 2019. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1395–1405.
[26] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*. 3307–3313.
[27] Mengying Wang, Hanchao Ma, Abhishek Daundkar, Sheng Guan, Yiyang Bian, Alpi Sehirlioglu, and Yinghui Wu. 2022. CRUX: Crowdsourced Materials Science Resource and Workflow Exploration. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 5014–5018.
[28] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 950–958.
[29] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.
[30] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *Conference on learning theory*. 25–54.
[31] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 235–244.
[32] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Junchi Yan, and Hongyuan Zha. 2021. Towards open-world recommendation: An inductive model-based collaborative filtering approach. In *International Conference on Machine Learning*. PMLR, 11329–11339.
[33] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.
[34] Yunfan Wu, Qi Cao, Huawei Shen, Shuchang Tao, and Xueqi Cheng. 2022. INMO: A Model-Agnostic and Scalable Module for Inductive Collaborative Filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 91–101.
[35] xray dataset [n.d.]. tolgadincer/labeled-chest-xray-images. https://www.kaggle.com/datasets/tolgadincer/labeled-chest-xray-images
[36] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*. PMLR, 12133–12143.