Selecting Top-k Data Science Models by Example Dataset

Mengying Wang
Case Western Reserve University
Cleveland, Ohio, USA
mxw767@case.edu

Sheng Guan
Case Western Reserve University
Cleveland, Ohio, USA
sxg967@case.edu

Hanchao Ma
Case Western Reserve University
Cleveland, Ohio, USA
hxm382@case.edu

Yiyang Bian
Haolai Che
yxb227@case.edu
hxc859@case.edu
Case Western Reserve University
Cleveland, Ohio, USA

Abhishek Daundkar
Case Western Reserve University
Cleveland, Ohio, USA
aad157@case.edu

Alp Sehirlioglu
Yinghui Wu
axs461@case.edu
yxw1650@case.edu
Case Western Reserve University
Cleveland, Ohio, USA

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 without 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, return k models that are expected to perform the best over the query dataset. (2) We formulate a regression problem and introduce a knowledge-enhanced framework using a model-data interaction graph. Unlike traditional methods, (1) ModsNet uses a dynamic, cost-bounded "probe-and-select" strategy to incrementally identify promising pre-trained models in a strict cold-start scenario (when a new dataset without any interaction with existing models is given). (2) To reduce the learning cost, we develop a clustering-based sparsification strategy to prune unpromising models and their interactions. (3) We showcase ModsNet built on top of a crowdsourced materials knowledge base platform. Our experiments verified its effectiveness, efficiency, and applications over real-world analytical pipelines.

CCS CONCEPTS

• **Information systems** → **Data management systems**; **Information retrieval**; **Top-k retrieval in databases**.

KEYWORDS

Model Selection; GNN-Based Recommendation; Knowledge Graph

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3615051

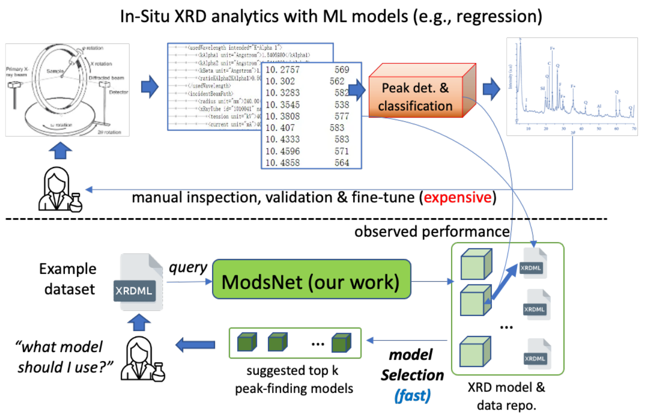


Figure 1: Selecting "Peek-finding" models to support peak analysis in X-Ray Diffraction Data (XRD).

ACM Reference Format:
Mengying Wang, Sheng Guan, Hanchao Ma, Yiyang Bian, Haolai Che, Abhishek Daundkar, Alp Sehirlioglu, and Yinghui Wu. 2023. Selecting Top-k Data Science Models by Example Dataset. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3615051

1 INTRODUCTION

The emerging data-driven analytical pipelines [19, 21] highlight the need to utilize various pre-trained *data science* models effectively. Unlike large-scale models with abundant training data from Web, such models are typically learned from small-scale, domain-specific datasets (*e.g.,* material science, biology, chemistry, physics), rely heavily on domain knowledge, and often take a great manual effort to tune due to high-dimensional in-situ configuration (*e.g.,* experiments, processing methods, devices access). In addition, validating the performance of data science 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 [10] without training a new model from scratch. Nevertheless, 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 [12], 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 X-ray Diffraction (XRD) datasets for peak-finding on Hugging Face. However, whether this particular model performs well or can be adapted with fine-tuning for new XRD datasets from another lab remains unclear.

Example 1: A 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 search by giving an XRD dataset as a representative "example". This would identify the top-k pre-trained peak-finding models for the "query" datasets, based on their historical performance over existing XRD datas. Ideally, this process should *avoid* expensive ad-hoc inference 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 validate the suggested models [19]. □

Such need is evident in promoting underutilized domain-specific models for accelerating scientific collaboration and improving the collaboration's trustworthiness. *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 [6] automate tuning pipelines over feature and parameter space. However, validating domain-specific models without domain knowledge remains expensive, especially given the large parameter search space.

(2) Several methods are proposed to select pre-trained source models for transfer learning [35]. These approaches often require re-training or inference tests, and rely on a "transferability score" that only quantifies the source-target relations. Instead, we recommend pre-trained models with good user-defined performances and eliminate costly testing, especially in 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 [8, 32], which can be categorized into three types: User-item extension, such as NGCF[27], lightGCN [11], and INMO-GCN [33]; Social network enhanced [7, 30]; and Knowledge graph enhanced[23, 26].

GNN-based methods [11, 27] usually learn bipartite user-item graph representations to predict missing links, assuming all nodes are observed. INMO-GCN [33] learns graph representation inductively without retraining but still requires test nodes to be in the graph by default. Social network-enhanced methods use the preferences of nearby users to improve user modeling. Knowledge-enhanced methods [23, 26] aim to enrich the node features with *e.g.,* metapath features from a knowledge graph. They all utilize link prediction and select items with high probabilities.

ModsNet is a knowledge graph-based model search system distinguishing the existing methods with the following. (1) ModsNet selects high-quality models with estimated good performances by a regression model instead of treating model search as a link prediction problem [23, 26]. (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) ModsNet reduces costs and maintains good erformance by using a clustering-based sparsification strategy to eliminate low-quality interactions from the large group of entities.

Model Selection for Transfer Learning. Transfer learning, in simple terms is knowledge transferring between similar task domains[17], 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[22], LEEP[15], and LogME[35]. 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.

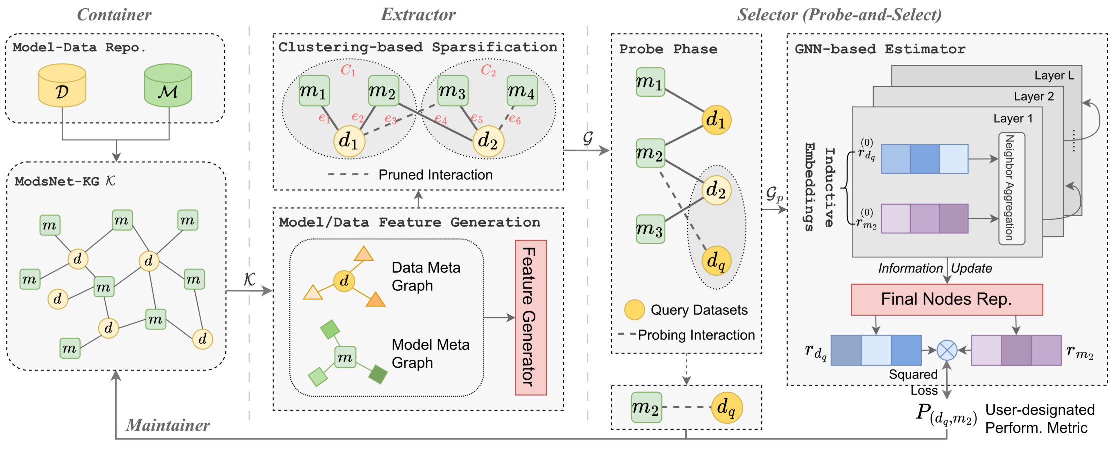


Figure 2: ModsNet Framework and Architecture

2 MODEL SELECTION: A FORMULATION

We start with several notations used by ModsNet framework.

Interaction Graph. ModsNet works with a *model-data interaction graph* $\mathcal{G} = (\mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{F})$, 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" [18] 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. We provide a list of examples ModsNet used in Section 4.

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 an interaction graph $\mathcal{G}$ simply as a node-induced subgraph of $\mathcal{K}$ with $\mathcal{M} \cup \mathcal{D}$. 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 data science models $\mathcal{M}$, datasets $\mathcal{D}$, and their interactions $\mathcal{I}$ with model performance 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 best performance in 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 $\mathcal{G}$.

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 (2θ), intensity ranges
- **Activity**: 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

(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 models, data and interactions for future model selection.

We next introduce the **Extractor** and the **Selector** modules in Sections 4 and Section 5, respectively.

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[14], BoW for textual attributes, and ast2vec[16] for script structures. The features can further be enriched by high-order path or topological features [26]. The enriched features are 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) similar models that differ from few (meta) features and are tested over a dataset should yield similar representations [13], and (thus) with comparable good performance;
- (2) a model is tested over similar datasets should contribute almost equally good performances;
- (3) if a model already performance well for some dataset, then its low-performance interactions over other datasets are less useful for model selection.

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 can be pruned 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 the above intuition, we introduce a clustering-based sparsification strategy to extract a small $\mathcal{G}$. The idea fully exploits the similarity of models, datasets, and their interactions to compute a set of quasi-bicliques in $\mathcal{G}$ that can *maximally* introduce the interactions in Cases (1) - (3).

Similarity measures. We first introduce several similarity measures. (1) We say two model nodes m, m' in $\mathcal{M}$ (resp. dataset nodes d, d' in $\mathcal{D}$) are *representationally* similar *w.r.t.* a threshold $\theta \in [0, 1]$, if $\mathsf{sim}\ (\mathcal{F}(m), \mathcal{F}(m')) \geq \beta_m$ (resp. $\mathsf{sim}\ (\mathcal{F}(d), \mathcal{F}(d')) \geq \beta_d$). Here sim is a similarity metric that quantifies the feature similarity of m and m' (resp. d and d'). In practice, sim can be defined as cosine similarity, centered kernel [13] or other metrics. (2) Given two interactions $e = (m, d)$, $e' = (m', d')$ and a threshold θ, we say e and e' are similar if $\mathsf{Isim} \geq \theta$. Here

$$\begin{aligned}\mathsf{Isim}(e, e') = \frac{\alpha}{2} \cdot ((\mathsf{sim}(\mathcal{F}(m), \mathcal{F}(m'))\\ + \mathsf{sim}(\mathcal{F}(d), \mathcal{F}(d')))\\ + (1-\alpha)\frac{|P(e) - P(e')|}{\max(P(e), P(e'))}\end{aligned}$$

In other words, two interactions are more similar, if they involve a pair of more similar model nodes and similar datasets, with comparable model performance.

Interaction Sparsification. We now characterize our sparsification strategy. Given an interaction graph $\mathcal{G} = (\mathcal{M}, \mathcal{D}, \mathcal{I})$, we say a clustering $C = \{C_1, \ldots C_n\}$ of the node set $\mathcal{M} \cup \mathcal{D}$ is *compatible* if (1) for any two model (resp. dataset) nodes $m, m' \in C_i$ (resp. $d, d' \in C_i$) for some cluster $C_i \in C$, m and m' (resp. d and d') are similar; and (2) for any two interactions (m,d) and (m',d') in $\mathcal{I}$, where m, m', d and d' are all in C_i, then (m,d) and (m',d') are similar. Let the set $C_{\mathcal{I}}$ be the interactions with both of its end nodes in some cluster, *i.e.,* $C_{\mathcal{I}} = \{(m,d) | m, d \in C_i, i \in [1,m]\}$. We define the *cost* of a compatible clustering as:

$$\text{cost}(C) = \Sigma_{(m,d) \in \mathcal{M} \times \mathcal{D}} X_{(m,d)}$$

where $X_{(m,d)}$ is a Boolean indicator for each pair $(m,d) \in \mathcal{M} \times \mathcal{D}$), and is defined as:

$$X_{(m,d)} = \begin{cases} 1, \text{if } (m,d) \in (\mathcal{I} \setminus C_{\mathcal{I}}) \cup (C_{\mathcal{I}} \setminus \mathcal{I}) \\ 0, \text{otherwise} \end{cases}$$

The problem of interaction sparsification, denoted as MDIM, is to compute a compatible clustering C with the smallest cost cost(C). One can verify that computing an optimal clustering with smallest cost is equivalent to maximize the prunable interactions that are "within clusters" (Case (1) and (2)), as well as those that are "cross clusters" (Case (3)), subject to the definition of compatible clustering. We present the following result.

Theorem 1: *Given an interaction graph $\mathcal{G}$, and similarity thresholds for $\mathcal{M}$, $\mathcal{D}$ and $\mathcal{I}$, (1)* MDIM *is* NP-*hard; (2) there is a* 4*-approximation algorithm for* MDIM *and runs in time* $O(|\mathcal{K}| + |\mathcal{M}||\mathcal{D}|)$. □

The hardness of MDIM can be verified by a reduction from the Exact 3-Set Cover problem, a known NP-complete problem [9] (see the detailed proof in [24]). We next outline an algorithm that first computes a clustering C with the quality guarantee, and then prunes $\mathcal{G}$ with induced interactions.

Algorithm. The algorithm, denoted as APXIM (illustrated in Fig. 4), induces the sparsified bipartite $\mathcal{G}_0$ from knowledge graph $\mathcal{K}$, by performing the following two phases. (1) In the clustering step, it initializes a single cluster C_{curr} by randomly selecting a model node m_s from $\mathcal{M}$ and adding m_s and its neighbors in $\mathcal{G}_0$ to a current cluster C_{curr} with m_s (subject to compatability constraint on dataset similarity - not shown) and removes m_s from M (lines 4-5); b) it then iteratively checks if the cost can be reduced by adding a new similar model node to C_{curr} (line 12), or instead asserts a new singleton cluster (lines 14-15). More specifically, APXIM estimates the impact of introducing a new node m with its neighbors to the cost (lines 7-9), and exploits a randomization selection strategy (line 11) to decide whether to include m to C_{curr}, with guarded conditions to ensure compatibility. This step simulates a randomized selection for maintaining a *bipartite correlation clustering* [3]. (2) Once a compatible clustering is formed, it invokes procedure Prune (with details presented in [24]) to keep a small sample of similar interactions within each cluster (Case (1) and (2)), and prunes interactions that have better counterparts (Case (3)).

Example 2: As illustrated in the Extractor module 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 in "clustering" phase (line 2-17 in Fig 4), (2) it then prunes interactions e_3, as e_3 and e_4 are two "cross-cluster" interactions bridging C1 and C2, where the performance of e_3 is lower than e_4. (3) It also prunes e_6, as sim(e_5, e_6) is found to be larger than a pre-defined threshold $\theta = 0.5$, and e_5 performs better than e_6. □

```
Algorithm APXIM
Input: knowledge graph $\mathcal{K}$, threshold $\theta$.
Output: a sparsified bipartite graph $\mathcal{G}$.
1.  attributed bipartite graph $\mathcal{G}_0$ := Induce($\mathcal{K}$)
    /* induce original interaction graph from knowledge graph $\mathcal{K}$ */
2.  set $C := \emptyset$; set $M := \mathcal{G}_0.\mathcal{M}$;
3.  while $M \neq \emptyset$ do
4.    model node $m_s$ := randomSelect($M$);
5.    set $C_{curr} := \{m_s\} \cup$ neighbor($m_s$); $M = M \setminus \{m_s\}$;
6.    for each $m \in M$ similar with $m_s$ do
7.      set $S_1$ := neighbor($m_s$) \ neighbor($m$);
8.      set $S_2$ := neighbor($m$) \ neighbor($m_s$);
9.      set $S_{1,2}$ := neighbor($m$) ∩ neighbor($m_s$);
10.     double $seed$ := random(0, 1);
11.     if $seed \in [0, min(\frac{|S_{1,2}|}{|S_{2,1}|}, 1]$ then
12.       if $|S_{1,2}| \geq |S_1|$ then $C_{curr} := C_{curr} \cup \{m\}$;
13.       else if $|S_{1,2}| < |S_1|$ then
14.         $C_{single} := \{m\}$;
15.         $C := C \cup C_{single}$;
16.       $M := M \setminus \{m\}$;
17.   $C := C \cup \{C_{curr}\}$;
18. $\mathcal{G}$ := Prune($C, \theta, \mathcal{G}_0$);
19. return $\mathcal{G}$;
```

Figure 4: Algorithm APXIM

Analysis. APXIM ensures a 4-approximation for MDIM. We verify this with an approximation preserving reduction from MDIM to a *bipartite correlation clustering problem* (BCC) [3]. Given a bipartite graph $G = (L, R, E)$, where L and R are the set of nodes; and E is the edge set of G. BCC computes a node clustering Cl of $L \cup R$, which are identified by a bipartite graph $B = (L, R, B_E)$ such that B_E only preserves edges within each cluster. It minimizes a symmetric difference between E and B_E, for which our cost measure exactly quantifies. As our algorithm simulates the randomized greedy selection strategy that ensures a 4-approximation ratio for BCC [3], the approximation ratio follows.

For the time cost, APXIM takes $O(|\mathcal{K}|)$ to initialize $\mathcal{G}$. It takes at most $|\mathcal{M}|$ rounds, and each round incurs a cost in $O(|\mathcal{I}|+|\mathcal{D}|)$ time to verify at least one model and its neighbors to decide whether to enlarge a cluster or to create a new one (lines 7-19). The pruning phase takes $O(|\mathcal{I}|)$ time. As $|\mathcal{I}| \leq |\mathcal{M}||\mathcal{D}|$, APXIM takes in total $O(|\mathcal{K}| + |\mathcal{M}||\mathcal{D}|)$ time to perform the interaction pruning.

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 tackling the *strict cold-start* scenarios (case(2)(b) in Section 3), and furnishes new factual knowledge for **Maintainer** into $\mathcal{K}$.

5.1 Probe and Select

As remarked earlier, a model tends to produce similar representations and performance over two datasets d and d', if d and d'

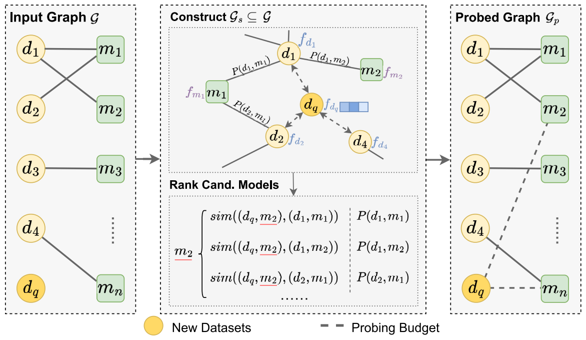

Input Graph $\mathcal{G}$
d_1
d_2
d_3
d_4
d_q
m_1
m_2
m_3
m_n
Construct $\mathcal{G}_s \subseteq \mathcal{G}$
f_{d_1}
$P(d_1, m_1)$
$P(d_1, m_2)$
f_{m_2}
f_{m_1}
$P(d_2, m_1)$
f_{d_q}
f_{d_2}
f_{d_4}
Rank Cand. Models
m_2
$sim((d_q, m_2), (d_1, m_1))$ | $P(d_1, m_1)$
$sim((d_q, m_2), (d_1, m_2))$ | $P(d_1, m_2)$
$sim((d_q, m_2), (d_2, m_1))$ | $P(d_2, m_1)$
Probed Graph $\mathcal{G}_p$
● New Datasets
- - Probing Budget

Figure 5: Probe Phase

are similar in terms of their representations [13]. Consistently following this intuition, we introduce a *Probe-and-Select* algorithm (illustrated in Fig. 5), which performs two steps: (1) Probe: for a new example dataset d_q (a "query"), we select a set of models $\mathcal{M}_{prob}$ and insert "probing" interactions $\mathcal{I}_{prob}$ to be verified; (2) Select: for each selected model m in (1), we predict its performance on d_q by the GNN-based **Estimator** and return the top-k ones, and synchronize $\mathcal{K}$ with new interactions. Below we outline the algorithm.

Initialization. The algorithm first induces a subgraph $\mathcal{G}_s = (\mathcal{M}_s \cup \mathcal{D}_s, \mathcal{I}_s, \mathcal{F}_s)$ of (sparsified) $\mathcal{G}$, where $\mathcal{D}_s$ refers to a set of l most similar datasets with d_q, $\mathcal{M}_s$ refers to the neighbors of $\mathcal{D}_s$ as candidate models for probing, along with induced interactions and features.

Probe Phase. Given $\mathcal{G}_s$, we rank candidate models with a score as:

$$Score(m) = \frac{1}{|\mathcal{I}_s|} \sum_{e \in \mathcal{I}_s} \mathsf{Isim}'((d_q, m), e) \cdot P(e)$$

Here Isim' is a variant of Isim with the second term ignored. That is, we rank the candidate consistently with the normalized interaction similarity as defined in Section 4.2, between (d_q, m) and each interaction e in $\mathcal{I}_s$. The only difference is that the second term in Isim is replaced by a "reward" term $P(e)$, as $P(d_q, m)$ is not known yet in the strict cold-start scenario. Indeed, the score favors the models for d_q that have good performance over similar datasets.

The algorithm then chooses the top b models with the highest scores from $\mathcal{M}_s$ as the probing models $\mathcal{M}_{prob}$, and then connect d_q with each model in $\mathcal{M}_{prob}$ to create probing interactions $\mathcal{I}_{prob}$. At this point, we get the updated 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. The top-k models with the best estimated performance are returned for d_q. The predicted results with the corresponding interactions are synchronized by **Maintainer** to enrich the $\mathcal{K}$.

Cost analysis. For strict cold-start, the probe and select phase takes $O(|\mathcal{D}_s| + |\mathcal{I}_s| + |\mathcal{M}_s| \log |\mathcal{M}_s|)$ time and $O(k * \mathsf{Inf} + k \log k)$ time for choosing top k models, where Inf is the cost of a single inference determined by $|\mathcal{F}|$, $|G_s| = |\mathcal{M}| + |\mathcal{D}|$ and the number of layers of the ModsNet model.

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 [33], 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:

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

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. α is used to dynamically control the degree of normalization. In the training phase, α gradually increases from 0.5 to 1. In the test phase, α is fixed as 1.

Graph Convolution Layers. ModsNet adopts typical graph convolution layers for GNN-based recommendations [4, 11, 27, 31]. Prior work shows that a trainable weight matrix has a negative effect, and nonlinear activation has no positive effect on recommendations. 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 [11].

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:

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

$\mathcal{L}_{\text{pred}}$ further adds a L_2 regularization term parameterized by λ to prevent overfitting. Θ 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.

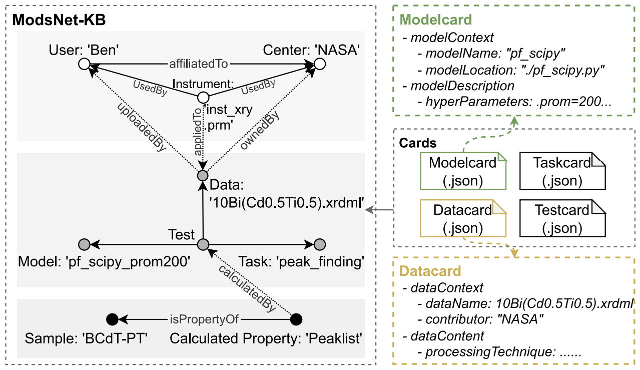


Figure 6: A fraction of ModsNet-KB

6 PROTOTYPE SYSTEM

As a proof of concept, we have built a prototype system of ModsNet. The system is supported by an established crowdsourced scientific knowledge graph system CRUX [25], which integrates data science models and datasets for material sciences. CRUX currently consists of a knowledge graph $\mathcal{K}$ with 462 data science models, 289 data repository including XRD datasets, 98,257 interactions, and in total 195,089 entities and 493,561 relations. The system uniformly stores $\mathcal{K}$ entities as JSON objects with a standardized format called "cards", specifying model cards, data cards, user cards, test cards (for interactions), and task cards (for task description), all managed in MongoDB. The datasets and model registration are supported by Google Cloud. A user-friendly query interface is developed to support "model search by example dataset". The registration and integration of datasets into $\mathcal{K}$ (**Maintainer**) is supported by an internal information extraction toolkit CRUX-IE [25].

7 EXPERIMENT STUDY

7.1 Experiment settings

We experimentally evaluate ModsNet with three real-world datasets. We investigate the following questions. (**RQ1**): The effectiveness of ModsNet in estimating performance of data science models over query datasets and in regression-based model selection. (**RQ2**): How well ModsNet perform with the varied amount of observed performances, the quality of data science 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. We also performed case analysis to show the real applications of ModsNet on peak finding and image classification.

Models, Datasets, and Knowledge Graphs. We gathered three real-world model-data repositories and constructed their knowledge graphs, including (1) KIZoo, which is a collection of Keras CNN models and image datasets for image classification sourced from Kaggle; (2) PKZoo, a repository of peak-finding models and XRD datasets from the crowdsourced materials knowledge base platform we developed and introduced in Section 6; and (3) HFZoo, which contains text classifiers, text datasets, and related information from Hugging Face [12]. See Table 1 for dataset details.

Model Selection Methods. We adapt 11 methods (including 3 variants of ModsNet) for model selection, categorized as follows:

(1) ModsNet *and its three variants*: ModsNet-NoKG is ModsNet without the knowledge graph equipped, it only relies on a weighted bipartite graph to make predictions; ModsNet-C is our method optimized by the clustering-based sparsification with high-quality interactions preserved; ModsNet-RProb randomly performs probes, without quality filtering for potential probing interactions. We use these variants to evaluate the ability of each module in Section 3.

(2) GNN-*based methods*: We use probe strategy from ModsNet to help three GNN-based methods handle the "cold-start" scenario. lightGCN [11], generates embeddings for collaborative filtering using linear propagation and a weighted sum function; IDCF-GCN [31], an inductive GCN framework that integrates matrix factorization and attention-based structure learning to predict links and their weights; INMO-GCN [33], which generates inductive embeddings for entities and reduces neighborhood bias through their interactions with template users/items.

(3) *Non*-GNN *collaborative filtering methods*: CF [29], predicts the rate matrix by interaction records and incorporates the new datasets based on the datasets' similarity; and Matchbox [20], which combined original CF and content-based filtering, solved the cold-start problem by leveraging the side information of the nodes. They always benefit greatly from dense interactions but with high delay.

(4) *Supervised learning methods*: LinearRegression, a simple linear regression model, and Wide&Deep [5], generalizes linear models with jointly learned deep neural networks to combine memorization and generalization for recommendation. These methods can solve cold-start problems and easily incorporate features into regression predictions, but struggle to capture high-order information.

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

For effectiveness, we report Precision@k, Recall@k, and NDCG@k, given a *relevant set* containing a set of pre-trained models with groundtruth performance higher than a specified threshold δ over the query datasets. This ensures that we fairly evaluate all the methods in identifying and suggesting "high-quality" models.

We control the number of interactions of model-data bipartite graphs with a factor θ, 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. We also evaluate the *Probe Strategy* by monitoring NDCG@10 with varying numbers of probes for GNN-Based methods. The default settings of k, θ, δ and number of probes 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 reported the mean.

7.2 Experiment results

Exp-1: Accuracy@k (RQ1). We report the performance of PKZoo and KIZoo in Tables 2 and 3, respectively, with a performance threshold of 0.8 for relevant sets.

(1) ModsNet outperforms all other approaches in all effectiveness metrics we recorded, regardless of the interaction density and the richness of the side information of the dataset.

Dataset	# models	# datasets	# interactions	# features	density	task	performance metric	# probes	k	δ	θ
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

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-C	0.887	0.841	0.108	**0.208**	0.888	0.862
ModsNet-RProb	0.867	0.733	0.112	0.186	0.859	0.777
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-C	0.72	0.72	0.158	0.322	0.705	0.707
ModsNet-NoKG	0.72	0.660	0.186	0.322	0.692	0.665
ModsNet-RProb	0.660	0.69	0.172	0.335	0.619	0.656
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

(2) We evaluate individual module's effectiveness by three variants: ModsNet-C achieved a comparable performance with ModsNet, yet with a sparsified model-data interaction graph with 31% interactions pruned, and reduced the learning cost of ModsNet by 22.85%. This verifies the effectiveness of the optimization. Also, the apparent gap between ModsNet and ModsNet-RProb highlights the advantage of the *Probe Strategy*, which means we accurately pick the high-quality probe interactions based on the interaction correlation we defined in Section 4. ModsNet-NoKG gets a poor score, while other methods here also incorporate the information from ModsNet-KG, which shows that the underlying knowledge graph provides strong support for this framework.

(3) Besides ModsNet, CF achieves good performance, in particular over PKZoo. However, its performance is not stable for other datasets. We observed that CF benefited greatly from the dense initial interactions over PKZoo, and it incurs much higher preprocessing costs, as will be discussed, which also makes it lose credits.

Exp-2:Impact of factors (RQ2). We next investigate the impact of the performance lowerbound δ, interaction ratio θ and number of probes to the effectiveness of ModsNet. We report the performances of the methods we introduced in Sec 7.1.

Varying δ. Fixing k = 10 and θ = 100%, we varied the threshold δ from 0.5 to 0.9 and reports the NDCG@10 in Fig. 7(a). All methods are getting worse, as it is harder to select higher-quality models, which will lead to a smaller relevant set. ModsNet is clearly ahead of others and exhibits stronger robustness across varying relevance thresholds, as our framework keeps identifying, preserving, and probing high-quality interactions. It also confirms the effectiveness of the probe strategy and knowledge graph, compared to ModsNet-RProb and ModsNet-NoKG variants.

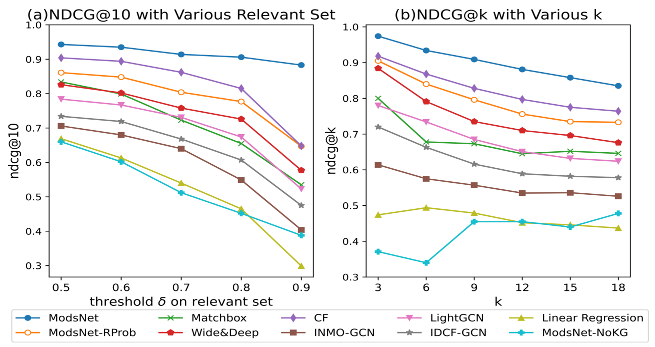


Figure 7: Effectiveness: impact of relevant set and k (PKZoo)

Varying k. As illustrated in Fig. 7(b), most of the methods perform worse as k varies from 3 to 18. As expected, selecting larger amount of pre-trained models allows ill-ranked models to contribute less in NDCG measures [28]. The performance of ModsNet is less sensitive to k, and remains the best among all the baseline methods. As k increases, ModsNet-NoKG's performance remains consistent due to its reliance on interaction performance alone. This may mean less accurate predictions, but they are stable.

Varying θ. We next evaluate the impact of the amount of available interactions. Fixing k = 10 and δ = 0.8, we varied θ from 20% to 100% and report the results in Fig. 8(a) and (b), over HFZoo and KIZoo. Most methods obviously benefit from dense interactions. While ModsNet illustrates robust performance over all cases, and outperforms other baselines in most cases as θ is larger.

Varying training set quality. We next investigate the impact of pre-trained models' quality in the training set. Fig. 8(c) depicts how NCDG@10 changes with the variation of the performance threshold δ on the training set from 0.4 to 0.9. It confirms that ModsNet maintains high-performance accuracy ($\sim$ 0.7) and effectively learns from pre-trained models with varying performance. Conversely, other approaches like Matchbox exhibit significant performance variations across different training sets, indicating the practical applicability of ModsNet, especially when model quality cannot be controlled or high-quality models are not always available.

Varying probe budget. We evaluate the effectiveness of the "probe-and-select" strategy over the number of interactions that are allowed to be performed, as shown in Fig. 8 (d). We only report the performance of GNN-based baselines, which need probes to cope with the "cold-start" issue. We observe that they do not always benefit from larger budgets as introducing too many less important nodes can decrease accuracy. Nevertheless, ModsNet maintains a higher NDCG@10 for a larger number of probes, and the decline occurs later, which shows our probe strategy accurately identifies the high-quality interactions to probing.

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

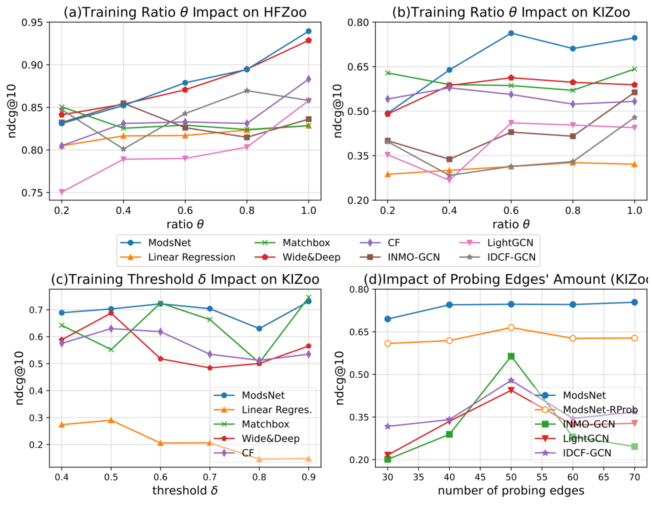


Figure 8: Impact of Factors

Training cost. Fig. 9(a) and (b) 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 25 seconds. As verified earlier, ModsNet-C further reduces the learning cost by 22.85% without sacrificing much in selection effectiveness. (2) As expected, All models take longer to train over more interactions. On the other hand, ModsNet provides the best results that outperform other methods, as verified earlier.

Query response time. We compare ModsNet and other methods with a query workload of 10 datasets and show the results in Fig. 9(c). (1) ModsNet is comparable with GNN-based approches. 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 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. Keeping the default setting as in Table 1, we varied the query workload size from 5 to 25. As shown in Fig. 9(d), ModsNet scales well with query workloads, and takes no more than 0.3 seconds to suggest all the answers. CF is the most sensitive method for various query loads.

Exp-4: Case Study. We next perform case studies to evaluate ModsNet with real 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 [34]) 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 XRD data. We present 2D data and ground truth peaks annotated by domain experts in Fig. 10. ModsNet recommends a pre-trained peak finding model with F_1 score of 0.807. In

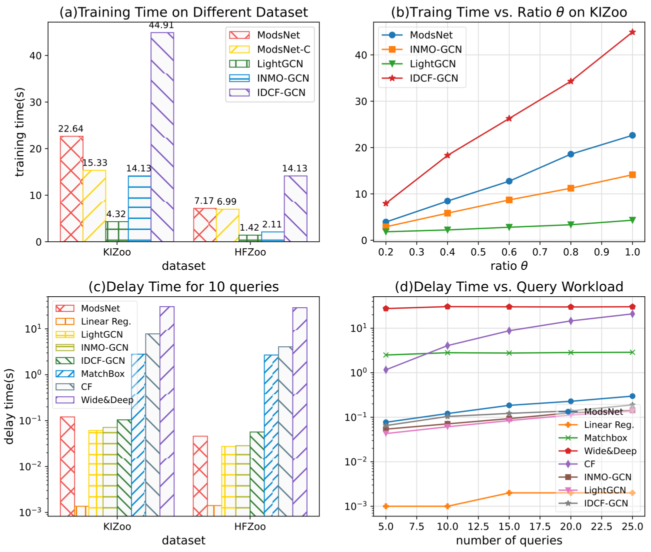
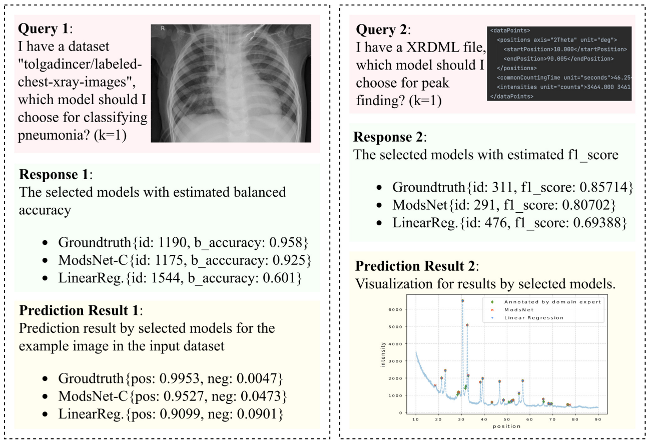


Figure 9: Efficiency of ModsNet



Query 1:
I have a dataset "tolgadincer/labeled-chest-xray-images", which model should I choose for classifying pneumonia? (k=1)

Response 1:
The selected models with estimated balanced accuracy

- Groundtruth{id: 1190, b_accuracy: 0.958}
- ModsNet-C{id: 1175, b_acccuracy: 0.925}
- LinearReg.{id: 1544, b_accuracy: 0.601}

Prediction Result 1:
Prediction result by selected models for the example image in the input dataset

- Groudtruth{pos: 0.9953, neg: 0.0047}
- ModsNet-C{pos: 0.9527, neg: 0.0473}
- LinearReg.{pos: 0.9099, neg: 0.0901}

Query 2:
I have a XRDML file, which model should I choose for peak finding? (k=1)

```
<dataPoints>
  <positions axis="2Theta" unit="deg">
    <startPosition>10.000</startPosition>
    <endPosition>90.005</endPosition>
  </positions>
  <commonCountingTime unit="seconds">46.25
  <intensities unit="counts">3464.000 3461
</dataPoints>
```

Response 2:
The selected models with estimated f1_score

- Groundtruth{id: 311, f1_score: 0.85714}
- ModsNet{id: 291, f1_score: 0.80702}
- LinearReg.{id: 476, f1_score: 0.69388}

Prediction Result 2:
Visualization for results by selected models.

Figure 10: Case study on application scenarios

contrast, the model selected by Linear Regression gets much lower accuracy than ours 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.

ACKNOWLEDGMENTS

This work is supported by NSF under CNS-1932574, ECCS-1933279, CNS-2028748 and OAC-2104007. Wu is funded in part by the U.S. Department of Energy under DE-EE0009353, and DE-NA0004104.

REFERENCES

[1] [n.d.]. Github. https://github.com/
[2] [n.d.]. Kaggle: Your Home for Data Science. https://www.kaggle.com/
[3] Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke Van Zuylen. 2012. Improved approximation algorithms for bipartite correlation clustering. *SIAM J. Comput.* (2012).
[4] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[5] 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 *DLRS*.
[6] Radwa Elshawi, Mohamed Maher, and Sherif Sakr. 2019. Automated machine learning: State-of-the-art and open challenges. *arXiv preprint arXiv:1906.02287* (2019).
[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*.
[8] 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. *TORS* (2022).
[9] Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness.*
[10] 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.
[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*.
[12] Hugging Face AI [n.d.]. Hugging Face – The AI Community Building the Future. https://huggingface.co/
[13] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *ICML*. 3519–3529.
[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[15] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. 2020. Leep: A new measure to evaluate transferability of learned representations. In *ICML*.
[16] Benjamin Paassen, Jessica McBroom, Bryn Jeffries, Irena Koprinska, Kalina Yacef, et al. 2021. Mapping python programs to vectors using recursive neural encodings. *JEDM* 13, 3 (2021), 1–35.
[17] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *TKDE* 22, 10 (2010), 1345–1359.
[18] 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.
[19] 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.
[20] David H Stern, Ralf Herbrich, and Thore Graepel. 2009. Matchbox: large scale online bayesian recommendations. In *WWW*.
[21] Jeyan Thiyagalingam, Mallikarjun Shankar, Geoffrey Fox, and Tony Hey. 2022. Scientific machine learning benchmarks. *Nature Reviews Physics* 4, 6 (2022), 413–420.
[22] Anh T Tran, Cuong V Nguyen, and Tal Hassner. 2019. Transferability and hardness of supervised classification tasks. In *ICCV*.
[23] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *WWW*.
[24] Mengying Wang, Sheng Guan, Hanchao Ma, Yiyang Bian, Haolai Che, Abhishek Daundkar, Alpi Sehirlioglu, and Yinghui Wu. 2023. ModsNet(Full Version). https://crux-project.github.io/assets/docs/ModsNet_Full.pdf
[25] 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 *CIKM*.
[26] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *KDD*.
[27] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*.
[28] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *COLT*.
[29] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2022. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *TKDE* 35, 5 (2022), 4425–4445.
[30] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *SIGIR*.
[31] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Junchi Yan, and Hongyuan Zha. 2021. Towards open-world recommendation: An inductive model-based collaborative filtering approach. In *ICML*.
[32] 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.
[33] Yunfan Wu, Qi Cao, Huawei Shen, Shuchang Tao, and Xueqi Cheng. 2022. INMO: A Model-Agnostic and Scalable Module for Inductive Collaborative Filtering. In *SIGIR*.
[34] xray dataset [n.d.]. tolgadincer/labeled-chest-xray-images. https://www.kaggle.com/datasets/tolgadincer/labeled-chest-xray-images
[35] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*.