ModsNet: Performance-aware Top-k Model Search using Exemplar Datasets

Mengying Wang[*]
Hanchao Ma[*]
Case Western Reserve University
Cleveland, Ohio, USA
mxw767@case.edu
hxm382@case.edu

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

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

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

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

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

ABSTRACT

We demonstrate **ModsNet**, a search tool for pre-trained data science **MOD**el**s** recommendatio**N** using **E**xamplar da**T**aset. Given a set of pre-trained data science models, an "example" input dataset, and a user-specified performance metric, ModsNet answers the following query: "*what are top-k models that have the best expected performance for the input data?*" The need for searching high-quality pre-trained models is evident in data-driven analysis. Inspired by "query by example" paradigm, ModsNet does not require users to write complex queries, but only provide an "examplar" dataset, a task description, and a performance measure as input, and can automatically suggest top-k matching models that are expected to have desirable performance to perform the task over the provided sample dataset. ModsNet utilizes a knowledge graph to integrate model performances over datasets and synchronizes it with a bipartite graph neural network to *estimate* model performance, reduce inference cost, and promptly respond to top-k model search queries. To cope with strict cold-start (upon receiving a new dataset when no historical performance of registered models are observed), it performs a dynamic, cost-bounded "probe-and-select" strategy to incrementally identify promising models. We demonstrate the application of ModsNet in enabling efficient scientific data analysis.

PVLDB Reference Format:
Mengying Wang, Hanchao Ma, Sheng Guan, Yiyang Bian, Haolai Che, Abhishek Daundkar, Alp Sehirlioglu, and Yinghui Wu. ModsNet: Performance-aware Top-k Model Search using Exemplar Datasets. PVLDB, 17(12): 4457 - 4460, 2024.
doi:10.14778/3685800.3685899

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/crux-project/crux-recommendModels.

[*]Both authors contributed equally to this research.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685899

1 INTRODUCTION

Emerging data-driven analysis [9] highlight the need to utilize various *data science models* effectively. Unlike large models with abundant training data, data science models are typically derived from small-scale, domain-specific datasets (*e.g.,* material science, biology, physics), and often take considerable manual effort to tune and validate due to large parameter space (*e.g.,* variables from experiments, processing methods, and device control).

One often prefers to reuse and fine-tune from existing high-quality models [5] without repeatedly rebuilding or testing from scratch. Data platforms such as Hugging Face gathered abundant scripts and datasets. Nevertheless, searching for models that are expected to perform well for a sample dataset remains to be a desired yet missing function for such data platforms.

ModsNet. We demonstrate ModsNet, a data engine to search for high-quality, underutilized pre-trained models for data-driven analysis. The system is an optimized, nontrival enhancement of our pilot study [10], with unique *new* features below.

"Search Models Simply with Data". ModsNet does not require users to write complex queries, but only needs a sample dataset (as a "query") to search for models. It jointly exploits "meta-features" extracted from datasets, models, and task description, and perform query-time performance estimation with a light-weighted graph neural network ("estimator") in support of online top-k search.

Performance-aware Search. Unlike existing data platforms that search models/scripts based on (exact) tag matching, ModsNet selects models with best expected performances over datasets and metrics of users' preferences. To this end, it is equipped with a knowledge graph that constantly synchronizes and accumulates factual knowledge about (estimated or validated) performances between models and datasets. This in turn improves the accuracy of performance estimation as new top-k queries are processed.

Test-free "Cold-start". ModsNet tackles *strict "cold-start"*, a known challenge in top-k selection, at query time. Queries may involve a dataset that have never been seen before, for which performance estimation is expensive due to estimator re-training, or re-testing

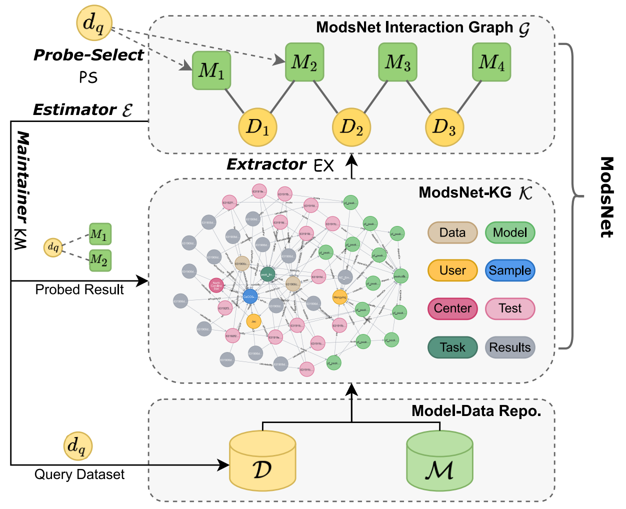


Figure 1: ModsNet Workflow: Overview

of models over the dataset. In response, ModsNet exploits a cost-bounded "probe-and-select" strategy that only explore models with promising performances over similar data, *without* retraining. This makes it feasible for large query workload and model repository.

On-Demand Visual Performance Analysis. ModsNet provides user-friendly interfaces to support fast access, search, and exploration of datasets and models. Users only need a "one-click" upload of exemplar dataset, and (optional) configuration to specify a number k and a task. ModsNet provides interactive panels for visual "card views", graph views, and on-demand performance test, to allow users to browse, explore, and evaluate the selected models.

2 FRAMEWORK OVERVIEW

We start with several notations and auxiliary structures used in ModsNet. These global structures are shared by all its components.

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 class of "model" nodes that refer to a library of user-registered data science models (or simply "models"), *e.g.,* a machine learning (ML) model, or a statistical model. (2) $\mathcal{D}$ is a class of "dataset" nodes, where each node $d \in \mathcal{D}$ refers to a dataset sample provided for targeted analytical tasks, *e.g.,* test data for ML models, simulation data, or measurements. (3) $\mathcal{I} \subseteq \mathcal{M} \times \mathcal{D}$ is a set of interactions (tests) edges. Each edge $(m, d) \in \mathcal{I}$ denotes an observed test of a model $m \in \mathcal{M}$ over a dataset $d \in \mathcal{D}$.

Meta-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" [8] such as statistics of model parameters, hyperparameters, training environment, test settings, etc. (3) Given an interaction $(m, d) \in \mathcal{I}$, $\mathcal{F}(m, d)$ also specifies 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 or training cost.

Knowledge graph. ModsNet-KG is a knowledge graph $\mathcal{K} = (\mathcal{V}, \mathbf{R})$ that curates factual knowledge of models, datasets and their interactions (test results and performances). 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 entities associated with models, such as contributors; and with datasets such as authors. These auxiliary entities enrich meta-features in $\mathcal{G}$.

Workflow. ModsNet consists of three major modular: (1) **Maintainer KM** tracks changes of repositories to synchronize a built-in knowledge graph $\mathcal{K}$; (2) **Extractor EX** maintains an interaction graph $\mathcal{G}$ with meta-features and updates a *GNN-based estimator model* $\mathcal{E}$ and (3) **Probe-Selector PS** selects models from $\mathcal{M}$ for query dataset by consulting the estimator $\mathcal{E}$ in limited times.

ModsNet will go through the following major steps for selection.

One-time Initialization. ModsNet "cold-starts" with a default knowledge graph $\mathcal{K}$. The extractor EX constructs an interaction graph $\mathcal{G}$. An estimator $\mathcal{E}$ is then trained as a link regression model over $\mathcal{G}$, to be consulted as a pre-trained model (see Section 3).

Model Selection. Upon receiving an exemplar dataset d_q (a query), ModsNet selects models for d_q with two cases:

(a) $d_q \in \mathcal{D}$, and there exists $m \in \mathcal{M}$ such that $(d_q, m) \in \mathcal{I}$, such as d_1 linked with m_1 and m_2 in Fig. 1. ModsNet will skip the *Probe Phase* process and invoke a standard transductive setting.

(b) Otherwise, if d_q is a new dataset without any existing interactions in $\mathcal{G}$, ModsNet will invoke *Probe-Selector* PS to identify limited models $\mathcal{M}'$ in $\mathcal{M}$ expected to perform well on d_q, then add probe edges by linking models in $\mathcal{M}'$ and d_q and make selections by consulting estimator $\mathcal{E}$ based on the probed graph (See Example 1).

Once d_q is processed, the Maintainer KM logs d_q, along with the probed tests, if any, as new interactions and enriches the knowledge graph $\mathcal{K}$ and interaction graph $\mathcal{G}$ accordingly. to keeps $\mathcal{K}$ "fresh".

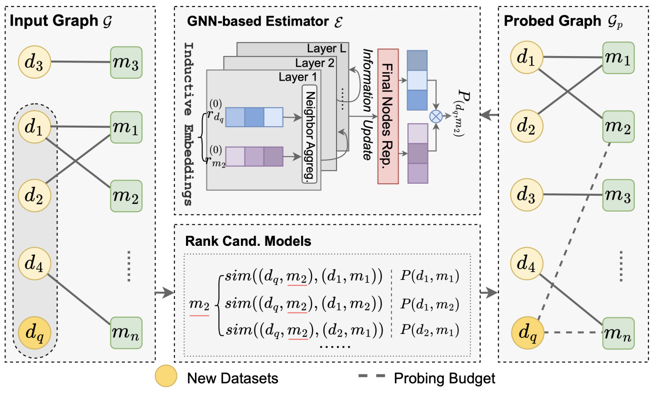
3 CORE MODELS AND ALGORITHMS

The demonstration will go through several fundamental models and algorithms that efficiently implement the three major modules.

Extractor. The Extraction module extracts and optimizes the bipartite Interaction graph $\mathcal{G}$ from the underlying knowledge graph $\mathcal{K}$. It mainly executes two major tasks: (1) feature generation, to enrich node features with auxiliary knowledge from $\mathcal{K}$; and (2) *sparsify* $\mathcal{G}$ by pruning unnecessary interactions (edges) between the model nodes and dataset nodes, to reduce the overhead of the training, performance estimation and recommendation cost.

Sparsifying Model-Data Interactions. To avoid pairwise tests, EX exploits a performance closeness measure by joining dataset similarity, model similarity, and the test results (*e.g.,* accuracy). It groups similar tests in terms of the performance closeness and conducts a sparsification process based on node-clustering over $\mathcal{G}$. This is based on the idea that *"similar models that only differ from few (meta)features and are tested over similar dataset should yield similar representations" [7], hence with comparable good performance."*

Performance Estimator. ModsNet trains a GNN-based Regression Model $\mathcal{E}$ to estimate the performance of a model m over a dataset d_p. It adopts graph convolution layers and train $\mathcal{E}$ with a



Input Graph $\mathcal{G}$
GNN-based Estimator $\mathcal{E}$
Probed Graph $\mathcal{G}_p$
Inductive Embeddings
Layer L
Layer 2
Layer 1
Neighbor Aggreg.
Information Update
Final Nodes Rep.
$P_{(d_q,m_2)}$
Rank Cand. Models
$sim((d_q,m_2),(d_1,m_1))$ $P(d_1,m_1)$
$sim((d_q,m_2),(d_1,m_2))$ $P(d_1,m_2)$
$sim((d_q,m_2),(d_2,m_1))$ $P(d_2,m_1)$
New Datasets
Probing Budget

Figure 2: "Probe-and-Select" Strategy

learning objective that minimizes a mean squared loss

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

where $\mathcal{I}_{\text{train}}$ refers to a set of historically observed tests in $\mathcal{G}$, $\hat{P}(d,m)$ and $P(d,m)$ refers to the predicted and estimated performance of the model m over dataset d, respectively; and $\mathcal{L}_{\text{pred}}$ includes a L_2 regularization parameterized by λ to prevent overfitting. Θ denotes trainable model parameters. Unlike traditional GCNs, ModsNet appends an inductive embedding generation layer on top of $\mathcal{E}$, inspired by a lightweight variant of GCN [12]. This allows ModsNet to quickly respond to unseen datasets *without* retraining.

Probe-Selector. The Probe-Selector module PS selects top-k models for a query dataset d_q by consulting **Extractor** EX with a bounded number of times. It performs two steps: (1) Probe: for a new top-k query d_q, it first samples a set of models $\mathcal{M}_{prob}$ with are likely to have high expected performance, and insert "probing" (virtual) edges to be verified; (2) Select: for each model m involved in the probing phase, it invokes the estimator $\mathcal{E}$ to predict model performance on d_q, which solves a link prediction task.

The models are then ranked based on a derived score, defined as

$$Score(m) = \frac{1}{|\mathcal{I}_s|} \sum_{e\in\mathcal{I}_s} \text{lsim}((d_q,m),e) \cdot P(e) \qquad (2)$$

Here $\mathcal{I}_s$ refers to the probed tests involving the datasets that are mostly similar to the query d_q, and the models having the highest performance; lsim is a similarity measure that is jointly determined by the dataset similarity, model representation similarity, and the estimated performance. Top-k models are then returned for d_q.

Example 1: As illustrated in Fig. 2, after obtaining an interaction graph $\mathcal{G}$ with embeddings of nodes and edges from *Extractor* EX, based on KNN clustering, ModsNet gets d_1, d_2, and d_4, which have the highest similarity to d_q, and retrieve models m_1, m_2, and m_n that linked with these datasets in $\mathcal{G}$ (left side). ModsNet then ranks the models (Equation 2) and selects the models (m_2 and m_n), which are assumed to perform well on d_q. It then creates two probe edges: (d_q, m_2) and (d_q, m_n) and get the probed graph $\mathcal{G}_p$ (right side). Finally, it consults GNN-based Estimator $\mathcal{E}$ to infer top-k models on d_q along with their estimated performance. □

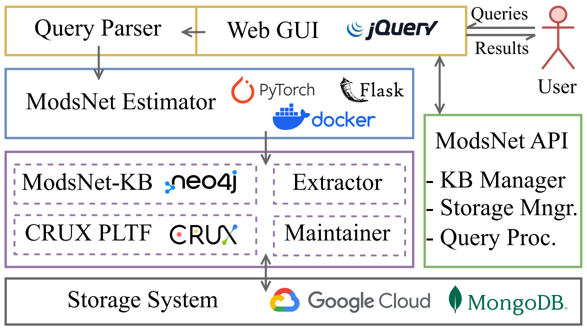


Figure 3: ModsNet **Architecture**

Time Cost. The once-for-all "cold-start" of ModsNet takes $O(|\mathcal{K}|+|\mathcal{M}||\mathcal{D}|)$ time to initialize the extractor. While using a L-layer GCN as an estimator $\mathcal{E}$ over $\mathcal{G}$ incurs an inherent cost of training and inference in $O(L|\mathcal{I}||\mathcal{F}| + L(|\mathcal{M}| + |\mathcal{D}|)|\mathcal{F}|^2)$ time [4], ModsNet optimizes a light-weighted GCN variant [6] that only incurs a once-for-all training of $\mathcal{E}$ in $O(L|\mathcal{I}||\mathcal{F}|)$ time, and a query-time inference in $O(|\mathcal{I}_s| \cdot |\mathcal{F}|)$ time. The query processing cost is in $O(|\mathcal{I}| + |\mathcal{M}|\log|\mathcal{M}|)$ ("probe-and-select") and in addition $O(k|\mathcal{I}_s| \cdot |\mathcal{F}| + k\log k)$ time for selecting top k models. We further verified that the sparsification effectively reduced on average 23% unpromising tests and speed up the training of $\mathcal{E}$ by 32%.

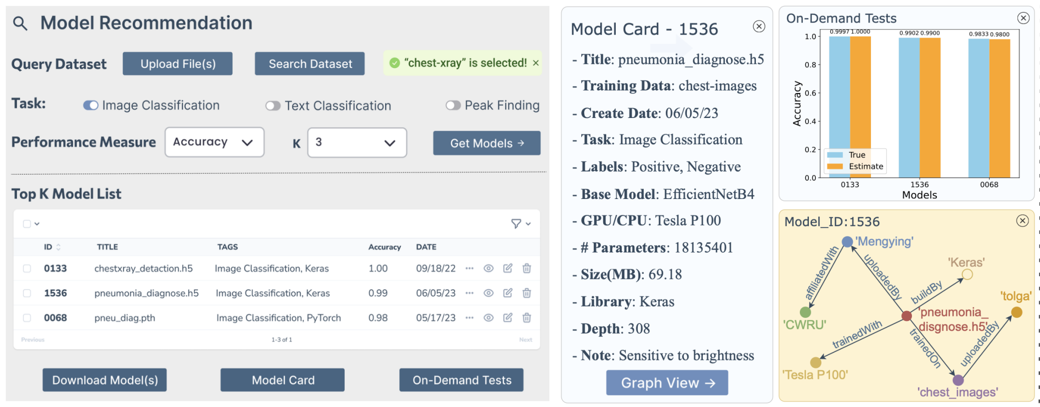
4 SYSTEM ARCHITECTURE

ModsNet is built upon a 4-layered design as illustrated in Fig. 3. (1) The *Query Layer* consists of (a) an interactive Web GUI with portals to collect crowdsourced datasets, models and their metadata, implemented with JQuery framework; (b) a query parser (supported by Flask APIs) to transform and deliver configuration and datasets to the extractor and maintainer modulars. (2) The *Estimation Layer* hosts a container that maintains the GNN-based estimator (trained with PyTorch). (3) The *Knowledge Management Layer* contains the core modulars: Extractor, Maintainer, and Probe-Selector. Along with the modular programs are also core component programs that coordinate data flow between ModsNet-KG hosted in Neo4j. The curation and synchronization of the knowledge graph is supported by an established crowdsourced platform [11]. (4) In the *Storage Layer*, ModsNet stores (a) the raw datasets $\mathcal{D}$, the model repository $\mathcal{M}$, and relevant metadata documents (in Google Cloud), (b) their JSON objects in the native MongoDB storage, and (c) indexed nodes and relations as triple stores of $\mathcal{K}$ in Neo4j. These data objects are cross-referenced with URLs, pointers, and indices, to support fast access. (5) ModsNet*APIs* support core functionalities in the modulars, such as data converters that transform raw dataset to featurized JSON objects, and interactive visualization.

5 DEMONSTRATION SCENARIO

Set up. We built a prototype system ModsNet [1][1] in JavaScript and Python. We demonstrate ModsNet with a client/server protocol,

[1]Beyond [10], the system has been significantly optimized with more comprehensive knowledge graphs, new built-in repositories, and APIs, optimized probing, and new on-demand test functions with visualization panels.



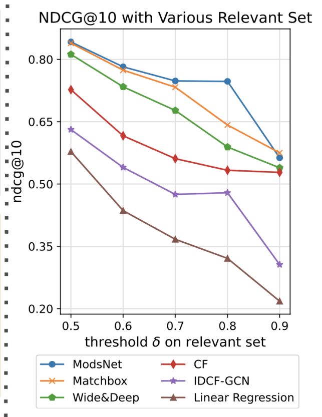
ID	TITLE	TAGS	Accuracy	DATE
0133	chestxray_detaction.h5	Image Classification, Keras	1.00	09/18/22
1536	pneumonia_diagnose.h5	Image Classification, Keras	0.99	06/05/23
0068	pneu_diag.pth	Image Classification, PyTorch	0.98	05/17/23

Model Card - 1536

- **Title**: pneumonia_diagnose.h5
- **Training Data**: chest-images
- **Create Date**: 06/05/23
- **Task**: Image Classification
- **Labels**: Positive, Negative
- **Base Model**: EfficientNetB4
- **GPU/CPU**: Tesla P100
- **# Parameters**: 18135401
- **Size(MB)**: 69.18
- **Library**: Keras
- **Depth**: 308
- **Note**: Sensitive to brightness

Models	True	Estimate
0133	0.9997	1.0000
1536	0.9902	0.9900
0068	0.9833	0.9800

(a) Visual Interfaces



(b) Evaluation

Figure 4: ModsNet: (a) Visual Interfaces, (b) Evaluation

with ModsNet hosted on a cluster of Linux cloud servers, each having an Intel Core i7 processor with 2.6 GHz and 16G memory.

Build-in Model Libraries. (1) KIZoo includes 1800 image classifiers, which accumulated over 1,000 GPU hours of training with various CNN architectures, involved 72 image datasets from Kaggle [3]. (2) HFZoo includes 932 text classifiers for 66 text datasets, collected from Hugging Face [2]. (3) PKZoo consists 462 models for "peak-finding" (a classification) task, 289 X-ray diffraction datasets, and 98257 tests. PKZoo is crowdsourced from a community of 6 institutions, with model accuracy validated by material scientists.

Query by Examplar Dataset. We invite users to experience an end-to-end, transparent top-k model selection session. Without writing any complex query, users only need to access a configurable, "one-click" dataset upload step, to find promising top k models. Users will also be able to download, inspect, and remove the datasets and models, which is hosted by registered independent query sessions. We go through a complete top-k search workflow, by introducing major modules and their interactions with knowledge graph, datasets, models, and their tests.

Performance-aware Selection. We shall invite users to compare the results and response time between ModsNet and several alternative selection algorithms. We report one result in Fig. 4(b) over PKZoo with 98257 tests. We fixed $k = 10$ and varied a threshold δ to constrain model selection from those with accuracy at least δ. We observed that ModsNet outperforms the other methods by 1.3 times on average in ranking quality (NDCG@10), with an average query response time at most 0.12 seconds. We will also show the impact of dataset size, model size, and k (omitted due to space limit).

Visual Performance Analysis. We will provide a thorough tour for users to the interactive panels for (1) "one-click" model and dataset upload, (2) configurable top-k search, and (3) on-demand tests and visual performance comparison. We illustrate the real-time monitor of the accuracy for the estimators. We will also guide users to experience the multiple visualized "model card" view and the graph views for user-friendly inspection of the returned models.

Applications. We showcase how ModsNet can help domain scientists accelerate the data analysis pipeline. (1) Our first scenario aims to select pneumonia diagnosis classifiers from KIZoo to determine whether a sample chest X-ray image indicates pneumonia-positive or not. ModsNet suggested top-3 models with the highest estimated accuracy in 0.1 seconds. On-demand tests verified that the estimated accuracy differ from actual counterparts with minor variances. (2) Other cases apply ModsNet for scientific text classification (using HFZoo), and peak finding for materials analysis (using PKZoo), respectively. In all scenarios, users can inspect the details of the selected models to decide proper adoption for downstream tasks.

ACKNOWLEDGMENTS

This work is supported by NSF under OAC-2104007.

REFERENCES

[1] 2024. Github. (2024). https://github.com/crux-project/crux-recommendModels
[2] 2024. Hugging Face – The AI Community Building the Future. (2024). https://huggingface.co/
[3] 2024. Kaggle: Your Home for Data Science. (2024). https://www.kaggle.com/
[4] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2020. Scalable graph neural networks via bidirectional propagation. *NeurIPS* (2020).
[5] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, and others. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250.
[6] 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*.
[7] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *ICML*. 3519–3529.
[8] 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.
[9] Jeyan Thiyagalingam, Mallikarjun Shankar, Geoffrey Fox, and Tony Hey. 2022. Scientific machine learning benchmarks. *Nature Reviews Physics* 4, 6 (2022), 413–420.
[10] 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 *CIKM*.
[11] 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*.
[12] 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*.