IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 October 2023, accepted 27 October 2023, date of publication 6 November 2023, date of current version 13 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3330647

RESEARCH ARTICLE

Metadata Representations for Queryable Repositories of Machine Learning Models

ZIYU LI[1], (Member, IEEE), HENK KANT[1], RIHAN HAI[1], (Member, IEEE), ASTERIOS KATSIFODIMOS[1], MARCO BRAMBILLA[2], (Member, IEEE), AND ALESSANDRO BOZZON[1]

[1]Department of Software Technology (ST), Faculty of Electrical Engineering Mathematics, and Computer Science (EEMCS), Delft University of Technology, 2628 CD Delft, The Netherlands
[2]Dipartimento di Elettronica Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy

Corresponding author: Ziyu Li (z.li-14@tudelft.nl)

This work was supported in part by Cognizant.

ABSTRACT Machine learning (ML) practitioners and organizations are building model repositories of pre-trained models, referred to as *model zoos*. These model zoos contain metadata describing the properties of the ML models and datasets. The metadata serves crucial roles for reporting, auditing, ensuring reproducibility, and enhancing interpretability. Despite the growing adoption of descriptive formats like datasheets and model cards, the metadata available in existing model zoos remains notably limited. Moreover, existing formats have limited expressiveness, thus constraining the potential use of model repositories, extending their purpose beyond mere storage for pre-trained models. This paper proposes a unified metadata representation format for model zoos. We illustrate that comprehensive metadata enables a diverse range of applications, encompassing model search, reuse, comparison, and composition of ML models. We also detail the design and highlight the implementation of an advanced model zoo system built on top of our proposed metadata representation.

INDEX TERMS Machine learning, metadata representations, model zoo, model search.

I. INTRODUCTION

Machine learning (ML) is increasingly used across application domains such as video analytics [1], [2], autonomous driving [3], content moderation [4], traffic monitoring [5] and crowd detection [6]. While ML models can be (and often are) trained for specific purposes, there is a growing interest in reusing and re-purposing of pre-trained ML models [7]. This shift, motivated mainly by computational, economic, and environmental reasons, is evident from the proliferation of public, pre-trained ML model zoos, such as HuggingFace, Tensorflow Hub, and PyTorch Hub.[1] These model zoos contain thousands of pre-trained models for diverse ML inference needs (e.g., recognition of classes/objects/concepts). Thanks to model zoos, complex predictive and analytics tasks can benefit from reusing existing ML models.

The potential of model zoos is currently hindered by the lack of *structured, comprehensive, and queryable* metadata representations. Current repositories include a wide range of information, e.g., using model cards [8]. However, such information is mostly for human consumption, and the level of detail remains coarse-grained, thus preventing advanced repository automation and management functionalities. TABLE 1 presents the information provided by different public model zoos. The categories of the information cover different aspects of ML artifacts (e.g., model, dataset, performance). We observe that current model zoos only provide limited information; for instance, PyTorch Hub provides only the ReadMe files from the source (e.g., a GitHub repository). Insufficient information forces practitioners to search for additional metadata in external repositories and descriptive documents or repeatedly go through the ML lifecycle. These processes impede the reuse of models and hamper their evaluation and assessment.

The software market provides several tools and platforms designed to help manage the ML lifecycle and

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan Kavak.

[1]https://huggingface.co/, https://www.tensorflow.org/, https://pytorch.org/hub/

TABLE 1. The existing public model zoos encompass various categories of metadata. The presence of metadata in each category can be denoted by symbols: ✓ indicates that the metadata is available, ∼ indicates that the metadata is available only in certain cases or provides limited information. Additionally, *queryability* is used to describe the type of queries that can be supported by the model zoo.

	Dataset Metadata				Evaluation Metadata		
Model Zoos	Source	Statistics	Configuration Metadata	Prediction Metadata	Metrics	Hardware specifications	Queryability
HuggingFace [9]	✓	∼	✓		∼		Faceted search
PyTorch Hub [10]			∼		∼		Faceted search
TensorFlow Hub [11]	✓		∼		∼	∼	Faceted search
Papers with Code [12]	✓				✓		Faceted search
OpenVino [13]	∼		∼		✓		
DawnBench [14]	✓		∼		✓	✓	Faceted search
Macaroni's Metamodel	✓	✓	✓	✓	✓	✓	Structured queries

experiment tracking with the support of metadata (e.g., MLflow [15], Weights & Biases [16], ClearML [17], comet [18]). However, the metadata is not standardized and requires custom formats from its users through API calls. While this approach may be convenient for users who wish to manage their private repositories, it falls short of facilitating broader knowledge sharing, integration, and reuse. This limitation ultimately hampers the full potential of leveraging metadata to enhance and optimize the ML lifecycle. Unlike these works focusing on managing the ML lifecycle, our work is specifically dedicated to addressing the metadata needs required to support model repositories and enable advanced functionalities, such as model retrieval and composition.

While data profiling techniques can determine metadata for datasets [19], the growing scope of ML applications (including their undesired effects on individuals and society [20]) requires metadata able to describe all the ML artifacts included in model zoos. The metadata should include information such as a model's inference capabilities (e.g., identified object classes), architecture, inference time, datasets (for training/validation/test), configurations (e.g., hyper-parameters values), and evaluation performance (refer to TABLE 3 for a complete list of metadata).

Such metadata can be helpful in different phases across the entire ML lifecycle [21], [22]. A few examples include: i) *Data cleaning and preprocessing*: The metadata of the training dataset can assist practitioners in identifying erroneous instances such that they can exclude the problematic data points during training [23]. ii) *Model selection*: Metadata accelerates the learning process by setting warm-starting of hyper-parameter searches instead of random search [24]. iii) *Model evaluation*: Metadata can facilitate keeping track of the performance of different objectives [7], as well as the predictions of each instance for further analysis (e.g., model explainability). iv) *Model explainability and reproducibility*: Metadata management can be used in the context of Trustworthy and Responsible AI.[2] Metadata can facilitate model explanation with model predictions and annotations [25]. v) *Model serving*: Metadata about model inference can perform model comparison and help practitioners decide which to use in production [26].

In addition, the availability of model metadata can also facilitate machine learning operations, commonly referred to as MLOps [27], and opens up new opportunities for advanced use cases such as i) retrieving models from large repositories with complex filtering conditions; ii) continuous integration of models in production (e.g., using transfer learning [28], [29]); iii) (semi-)automatic model composition; and iv) advanced model management system. We will further discuss the use cases in Section III-D.

In this work, we advocate for expressive metadata representation for model zoos. Beyond the current state-of-the-art (and practices) [30], [31], we propose a metadata format that can capture information about relevant artifacts (e.g., models, datasets, data instances, training configurations, evaluation) and their relationships. We also describe the design and current implementation of an advanced ML models management platform called *Macaroni* [32][3] that can be used to query and make use of such metadata. As seen in TABLE 1, our proposed metadata representations cover various categories of metadata and can support the querying of it, as in the example in TABLE 2.

The contributions of this paper are the following:

- We introduce a structured and queryable metadata model designed to provide comprehensive representations within model zoos, as detailed in Section IV.
- To validate our metadata model, we present *Macaroni*, a reference tool offering retrieval and analytical functionalities, operating across several model zoos. These functionalities are explained in Section V.
- Our work demonstrates how the metadata model facilitates automatic model reuse and composition. This is achieved through Boolean expressions over inference predicates and performance constraints, as outlined in Section VI.

[2]https://partnershiponai.org/paper/responsible-publication-recommendations/

[3]Prototype available at https://sites.google.com/view/macaroni-model-zoo/home

II. RELATED WORK

Recent studies focus on different aspects of management during the ML lifecycle, from model versioning, and model reporting to model evaluation. Each is important for practitioners to manage and understand the models. We observe a gap among the profound works [22], a comprehensive and queryable metadata representation. With the metadata representation, we can thus better manage the ML models and data, including the interactions between them.

A. ML MODEL MANAGEMENT SYSTEMS AND TOOLS

Due to the complexity of the ML models and ML lifecycle, managing the ML models in different phases are challenging. And yet, multiple systems have been developed to tackle the challenge of managing the ML models during training in experiments.

Modeldb [33] is one of the first systems that allow tracking, storing, and exploring ML models. Modeldb keeps track of the ML pipelines defined by the users and allows them to visualize and explore the models and pipelines. Other systems, such as ModelHub [34], ModelKB [35], and Runway [36] also allow managing ML experiments and their associated models. These systems allow model storage, versioning, and querying, with metadata being extracted from scripts or manually logged. None of them, unfortunately, made it explicit what metadata should be included/tracked during the experiments or when serving the models. Enterprises platforms, such as MLFlow [15], Amazon SageMaker, Google TFX [37], comet [18], Airbnb Bighead [38] and etc., provide rich APIs and tools to support ML experiment management. Users can customize what metadata to log by calling APIs. The metadata can be visualized or used to specify new experiments. In this work, we do not cover the aspect of managing ML model training experiments. Related works, e.g., [15], [16], [37], can be served as support and complement to our scope. We report models with rich and comprehensive metadata covering different artifacts and their relationships. We strive to support practitioners with the necessary information to know about a model and its necessary components.

B. ML INFERENCE/SERVING SYSTEMS

Instead of managing the end-to-end process of the ML lifecycle, multiple ML systems aim at a particular phase in the lifecycle, e.g., ML inference or ML serving.

1) ACCELERATING ML INFERENCE

Systems, such as Clipper [39], Willump [40], and GATI [41], optimize and accelerate ML inference when serving. The goal of these systems is to serve and infer ML models for downstream tasks. Clipper is a general-purpose low-latency prediction serving system that sits between end-user applications and a wide range of machine learning frameworks. It introduces a modular architecture to simplify model deployment across frameworks and applications. Clipper reduces prediction latency and improves prediction throughput, accuracy, and robustness without modifying the underlying ML frameworks.

2) ML BENCHMARK IN SPECIFIC DOMAINS

Researchers have been building ML benchmarks for different domains, for example, PMLB [42], PennAI [43](biomedical and health), Moleculenet [44] (molecule), facies classification [45], DLHub [46] (science), Kipoi [47] (genomics).

C. AI-CENTRIC DATA MANAGEMENT SYSTEMS

Systems have been developed and built to manage data for AI. DescribeML [48] and Amalur [49] propose dataset models to describe ML datasets in detail and preserve relevant metadata. The preservation of dataset information greatly facilitates, for example, the search for suitable datasets for ML projects. For ML dataset management and versioning, research work such as Mldp [31], Chimera [50], and DataLab [51] are complements to the above-mentioned model management systems that served as support for managing data versions.

Another type of data platform is to move the DBMS engine from a relational to a tensor abstraction, which unseemly integrates databases with external ML tools. TDP [52] provides access to multi-model data and leverages PyTorch to run queries over data on a wide range of hardware devices. TDP integrates the flexibility of PyTorch's programming model with the declarative power of SQL.

D. MODEL CARDS AND DATA SHEETS

Recent research also focuses on the reporting of models and datasets, covering aspects not only limited to basic informative components but also including ethical, inclusive, and fair considerations. Model cards [8], for example, proposed to include information regarding model intended use cases, potential pitfalls, and other contexts that can improve model understanding. A similar idea also lies in data cards/sheets. Examples include [53], [54], and [55]. Though model cards and data cards contain rich information, the Q&A format is nonetheless unfriendly to machines to process and thus cannot be easily managed and retrieved.

E. MODEL PERFORMANCE BENCHMARKING

A growing body of published work also focuses on the benchmarking of ML model performance, such as MLperf [56], [57], fathom [58], and DAWNBench [14]. These platforms covered a set of metadata, including metrics, and training and inference configurations with specified hardware/software settings. Their focus is the report of the model performance at different ML lifecycle stages (training or inference). They paid little attention to the dataset the model used, whose path is provided as an argument filled by the user. The model process pipeline is also not covered besides the model scripts.

III. ML MODEL ZOOS

The purpose of a model zoo is to store and provide access to different artifacts – and their descriptions – created throughout the lifecycle of ML. In this section, we first briefly describe the ML lifecycle, highlighting relevant artifacts. We then describe the organization and functionalities of model zoos.

A. MACHINE LEARNING ARTIFACTS

In the proposed metadata model, we tackle relationships among the artifacts in the entire ML lifecycle, including data collection, model training, model inference, serving, and reporting. We include the following artifacts that we believe should form the pillars of a rich model zoo: i) *model*, ii) *dataset*, iii) *configuration*, iv) *prediction* - semantic capabilities, v) and *performance*.

1) ML MODEL

Throughout the years, we have witnessed the advances of ML, and new models are developed with performance even surpassing human-level capabilities in many real-world tasks. Compared to the traditional ML models, such as regression models and decision trees, deep neural networks are far more complicated due to their complex architecture and their large number of parameters (some model sizes going beyond billions or trillions of parameters). To differentiate one model from others, simply knowing its name is insufficient. Additional information needs to be provided, not only for model reporting and reproducibility purposes but also for explainable AI.

2) DATASET

In recent years, there has been a notable emergence of the Data-centric discourse within the ML community, as elucidated in the study by Miranda et al. [59]. This emphasizes the crucial role of datasets in the training and testing of ML models. Datasets are crucial in ML for training models, evaluating performance, addressing biases, guiding feature selection, supporting transfer learning, assessing generalization and robustness, and promoting reproducibility and transparency. They serve as the foundation for developing effective and trustworthy ML models. Especially, understanding the data source and the collection methods becomes essential when diagnosing or debugging a model; yet, unfortunately, it is a frequently overlooked aspect in dataset reporting [55].

3) CONFIGURATION

A set of important configurations for model training are hyperparameters. By carefully selecting and tuning the hyperparameters, practitioners aim to optimize the model's ability to learn patterns from the data, reduce overfitting, and achieve better generalization to unseen samples. Besides hyperparameters, the configuration settings for ML model training or inference also include hardware configurations, such as CPUs, GPUs, or specialized accelerators. The hardware configurations not only affect the performance on latency measurement [60] but also on accuracy [61].

4) PREDICTION

Training a ML model is to fulfill a specific task for a real-world problem, e.g., image classification or named-entity recognition. Usually, the predictions are associated with the labels of the tasks. Researchers nowadays are investigating the semantic meaning of the predictions as concepts [62] to debug or explain the capability of a model. We highlight that the prediction outputs of a model with associated semantics play a significant role in explainable AI [62].

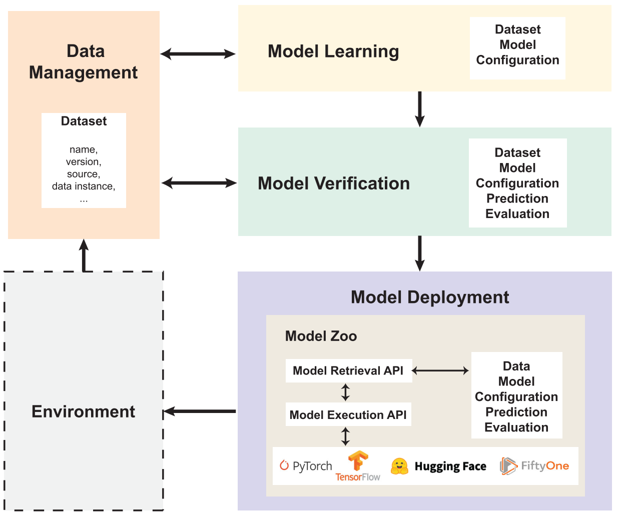


FIGURE 1. ML lifecycle along with metadata of different artifacts. We highlight the model zoo in Model Deployment phase with detailed components.

5) EVALUATION

The most straightforward way to observe the capability of a model is to evaluate it on a particular task. The majority of current deep learning performance benchmarks only measure the aggregated performance of the model, such as overall accuracy or processing time for a single minibatch of data. ML model performance is much more complicated in practice. Recent works, such as DAWNBench [63], propose measuring ML models' performance from diverse perspectives, including training time, inference latency, accuracy, etc. Besides these measurements, our proposed metadata system for model zoos also presents the per-class performance and supports a broader range of models.

B. MACHINE LEARNING LIFECYCLE

The ML lifecycle has no fixed definition. It is customizable and depends on the application and model. In the following, we will describe a general ML lifecycle based on [22], [33], [64], [65], and [66]. In general, the ML lifecycle includes four stages (sometimes some stages will be divided and regarded as separated stages): i) data preparation and data management; ii) model learning; iii) model evaluation and model verification; and iv) model deployment.

We now describe each of these phases to highlight which artifacts (and their properties) require consideration in the context of managing model repositories.

1) DATA PREPARATION AND DATA MANAGEMENT

The first stage of the ML lifecycle relates to the acquisition and transformation of data used by the ML models. This stage can be split into the following steps. i) Gather data samples through observations or measurements. ii) Analyse data: need for additional data, augmentation, and preprocessing. iii) Clean data: replace or remove incomplete data from the dataset. iv) Preprocessing: convert raw data such that ML models can use it. v) Feature engineering: extract relevant features from raw data such that it can be used for model building. vi) Separate data into training, validation, and test sets. A model zoo should use metadata that is able to capture information about data preparation and processing (e.g., data source, data curation method, data statistics), as such data-related operations influence the performance of a ML model [67].

2) MODEL LEARNING

The *model learning* stage concerns the design and training of the ML model. This stage can be divided into the following steps: i) model selection depending on the type of data (structured or unstructured); ii) selection of the loss function to measure training error; iii) selecting and tuning hyperparameters to control overfitting, underfitting, and other characteristics; iv) model training or finetuning on datasets to minimize error; v) repeat steps 3 and 4 until good precision numbers and low training error. Capturing metadata about the configurations, such as model architecture, hyperparameters, etc., is important to understand and interpret the performance of a model.

3) MODEL EVALUATION AND MODEL VERIFICATION

After the model training, the model needs to be verified on unseen (validation or testing) data. The main goal of this stage is to ensure that the model, after training, performs as expected on new inputs. Usually, this is done by assessing the performance of the trained model against a test dataset that was generated in the data management stage. The relevant metadata regarding the dataset and model shall be captured. In addition, the relevant configurations, such as hardware settings, should also be recorded.

4) MODEL DEPLOYMENT

The outcome of this stage is a properly functioning, fully-fledged, and deployed ML system. At this stage, all the information from previous stages shall be revealed, such as dataset information, model training details, and, most importantly, the performance under different environment specifications. The environment specifications include hardware specifications and any specific software configurations or constraints. Fine-grained metadata helps ensure compatibility and optimal performance of the deployed model. Practitioners can thus choose the appropriate models for their needs and requirements.

C. EXISTING MODEL ZOOS

Recently, communities have been focusing on democratizing ML, for both using and sharing. Platforms, often referred to as model zoos, such as Tensorflow Hub, PyTorch Hub, and HuggingFace, are now exposing metadata related to ML artifacts. These platforms/hubs are building on the principles of using open-source model parameters, scripts, and APIs. HuggingFace also offers an abstraction with the *Transformers* library, which makes it easy to consume and infer these models [9]. In the following, we describe the capabilities and limitations of currently available model zoos.

TABLE 2. Example queries.

Property	ID	Query
Dataset information	1	Retrieve text classification models trained on dataset crowdsourced by at least a group of 50 people
	2	Find a dataset collected from COCO and OpenImage with all the images containing “dog”
Model performance	3	Retrieve models trained on ImageNet with an accuracy higher than 90%
	4	Which model performs the best on COCO for person detection?
Interpretability	5	Retrieve a person detection model with no gender bias
	6	Retrieve text generation models that do not generate hate speech
Hardware-related	7	Retrieve image classification models that are suitable to deploy on edge devices

1) METADATA

Existing model zoos offer users model cards [8] that describe the models in different levels of detail. Model cards contain metadata regarding different artifacts: For example, in TensorFlow Hub, they include metadata such as model publisher, architecture, data that train the model, inputs and outputs. Pytorch Hub includes model description, usage, and results. Besides the mentioned metadata, HuggingFace also includes discussions on the bias and limitations of the model. The metadata in these model zoos provides practitioners with different aspects of the artifacts.

2) FUNCTIONALITIES

Given the metadata, model zoos provide access to retrieving the models by means of filtering by name, task, or trained data. Some of the model zoos also support data retrieval given provided metadata of the related dataset. On top of all functionalities, model inference and sharing is the most important feature of these model zoos. They provide corresponding APIs for model execution. Practitioners can consume these public ML models for downstream tasks, e.g., building applications on top of the models, and finetuning the models on the specified dataset. For example, the APIs of HuggingFace are built on top of transformers [68], and users can easily infer models in NLP domains.

3) LIMITATIONS

Model zoos differ in the type and level of detail for the metadata associated with the different ML artifacts.

Therefore, practitioners will need to manually retrieve and integrate information from various platforms if the same artifact (e.g., a dataset) is hosted in multiple zoos. Metadata quality and consistency are also an issue: often, models are described with little or no metadata; for example, it is very common for users of the HuggingFace platform to upload only the trained models without additional descriptions. Finally, metadata are often not offered in a computer-readable format: most of the descriptions of the artifacts are presented in plain texts, created only for human consumption. Obviously, this greatly hinders the retrieval capabilities of the model search engines and the ability of practitioners to compare different models.

D. REQUIREMENTS FOR FUTURE MODEL ZOOS

With the existing public model zoos, we observe some limitations that hinder the reuse/sharing/management of ML models. We foresee that the future model zoo should contain one of the following attributes: i) rich metadata representations that enable querying the repository; ii) rich analytic capabilities to facilitate advanced performance evaluation and comparison; iii) offering different functionalities, including serving the models through APIs or endpoints, and integrating the downstream systems that (e.g.,) accelerate model inference.

1) RICH METADATA REPRESENTATIONS

The model zoo should contain information regarding different artifacts, e.g., data, ML model details, model performance, etc. This information allows practitioners to be aware of how the model is defined, on what dataset it is trained, and the corresponding evaluation performance. With rich metadata representation, practitioners may not only have access to comprehensive information but also search/query on top of it, which enables data/model search, data/model discovery, and comparison. TABLE 2 lists some example queries that could be valuable for ML developers and users. These queries require more fine-grained model metadata that current model repositories, such as HuggingFace, do not support. This highlights the need for detailed metadata of trained ML models and datasets in a structured and queryable representation.

2) OPERATIONS THAT FACILITATE ADVANCED PERFORMANCE EVALUATION

A model zoo is not a mere information presentation platform, but it shall also serves as a tool for practitioners to facilitate advanced performance evaluation. When a new/updated dataset is provided (provided by the practitioners or added by the system), the model zoo shall allow automated evaluation/finetuning or provides operations for the practitioners to evaluate/finetune models on top of it. For example, a practitioner would like to test the robustness of a model by evaluating the model performance on a perturbed dataset. To facilitate this, the model zoo shall first allow operations on a dataset for perturbations (e.g., adding noise, adversarial attacks) or users uploading their own data. In addition, the model zoo shall also provide APIs to perform the evaluation of the dedicated models on the perturbed dataset.

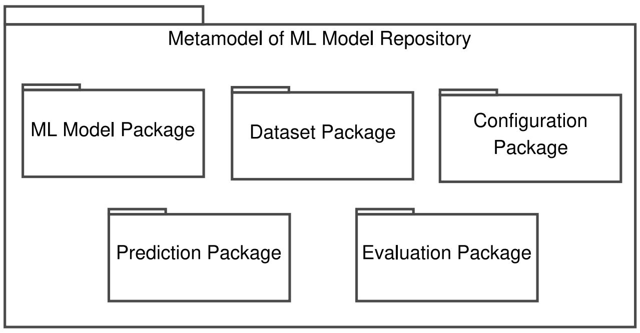


FIGURE 2. The Model Zoo Metamodel.

3) OTHER FUNCTIONALITIES

The purpose of having a model zoo includes sharing and serving a model. The model zoo shall also provide easily-access APIs or endpoints to serve/deploy a model. With models of various characteristics (e.g., tasks to answer, evaluation performance), the model zoo makes it feasible to solve complex analytic tasks by constructing workflows through the composition of models. Optimizations can focus on how to define the workflow under requirement constraints (e.g., accuracy or latency) [69], or how to assign workload on heterogeneous hardware (e.g., edge or server) [70].

IV. PROPOSED METAMODEL

Based on the analysis of the current capabilities and limitations of current model zoos, we now describe the metadata format (i.e., the *metamodel*) that can be used to represent different ML artifacts and their relations. With our proposed structured representation along with comprehensive metadata, users can retrieve metadata in fine-grained details.

FIGURE 2 depicts the main sub-models that compose our *metamodel* in a bird's eye view. The *metamodel* comprises five packages:

i) the `ML Model` package, which defines the ML models, their architecture, input, and output formats;
ii) the `Dataset` package, which contains the information on the datasets;
iii) the `Configuration` package, which summarizes the configuration settings when training and using a model for inference;
iv) the `Prediction` package, which describes the inference output of the model, possibly enriched with description from a knowledge graph;
v) the `Evaluation` package, which presents the different evaluation metrics, including the ones related to output accuracy and to time performance.

TABLE 3 shows the summary of the metadata included in our metamodel, and the associated artifacts.

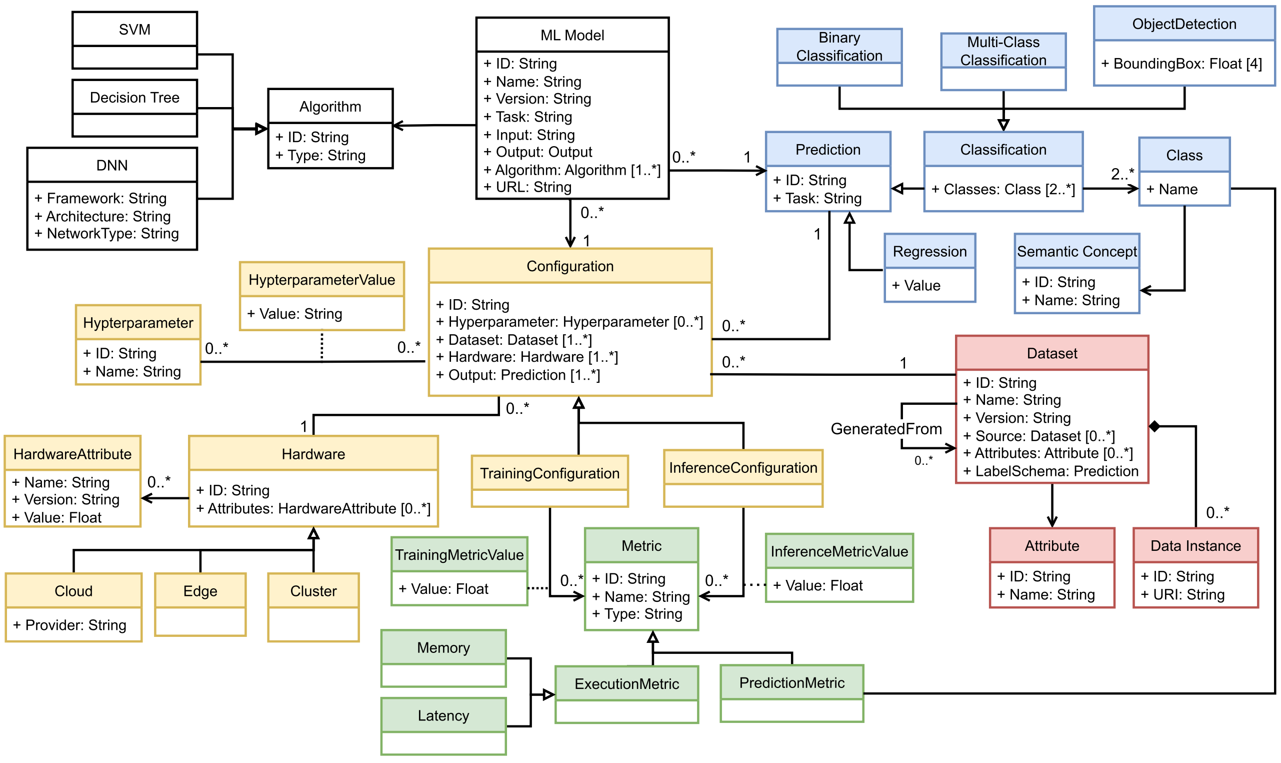


FIGURE 3. Modeling the metadata throughout ML lifecycle.

TABLE 3. Metadata summary.

Artifacts	Metadata type
ML Model	ID, name, version, task, architecture, input & output format/size, source URL, algorithm (e.g., SVM, decision tree, DNN)
Dataset	ID, name, version, source, label schema, data instance ID, data instance attribute, data instance URI
Configuration	ID, configuration type (training/inference/testing), hyperparameter name, hyperparameter value, hardware attribute version
Prediction	ID, task, classes (classification task), values (regression task), bounding box (object detection task), semantic concepts
Evaluation	evaluation type, evaluation metric name, evaluation value

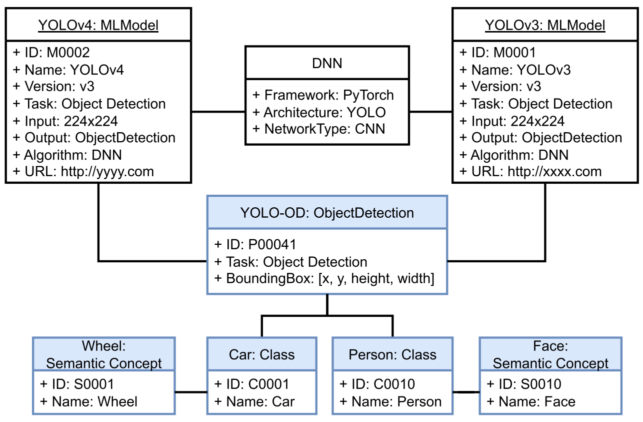


FIGURE 4. ***ML Model*** **and** ***Prediction*** **model extract for a running example.**

A. ML MODEL PACKAGE

The taxonomy for the `ML Model` package, depicted in FIGURE 3, encompasses classes with a white background, delineating various components such as ML Model basic information and the algorithm. The metadata associated with the `ML Model` encompasses essential details such as name, version, tasks, input and output specifications (I/O), as well as a URL linking to the script files. The `algorithm` specify the ML algorithms in different categories: traditional ML algorithms, such as SVM, decision tree, and Deep Neural Network (DNN). DNN can be further divided into different types of networks, e.g., CNN and RNN. One of the trends of advancement in DNNs is characterized by continuous innovation and the development of increasingly sophisticated architectures, e.g., transformers [68], GPT [71], BERT [72]. Recording the framework and architecture of these advanced models is thus fundamental. The availability of such metadata is of importance in facilitating model management, model understanding, and interoperability within the ML ecosystem, and it promotes trust and confidence in the models.

B. DATASET PACKAGE

The behavior of a ML model heavily relies on the data that has been used for training it. Thus the *metamodel* includes a Dataset package, representing both Datasets and their *DataObject*s. With the Dataset element, we present the metadata of the datasets that is significant for data management and reporting. Examples of the metadata are i) *ID*, to uniquely refer to a dataset; ii) *Name*, name of the dataset; iii) *Version*, version of the dataset, e.g., COCO has multiple versions constructed in different years; iv) *Source*, reference to other dataset(s) which (partially) construct the current dataset; v) *Attributes*, the attributes contained in the dataset, specifically for structured datasets, specifying different columns.

A dataset consists of multiple data objects. With DataObject, we denote the i) *ID*, to uniquely refer to a piece of content; and ii) *URI*, a string that unambiguously identifies the location of the content.

FIGURE 5 shows the datasets applied in the example. COCO is a popular image dataset that is usually used to train object detection and image segmentation models. In this case, we split COCO into a training set and a testing set. Both datasets have the same source, which is the complete COCO dataset. The COCO training set and test set contain a different subset from the complete dataset.

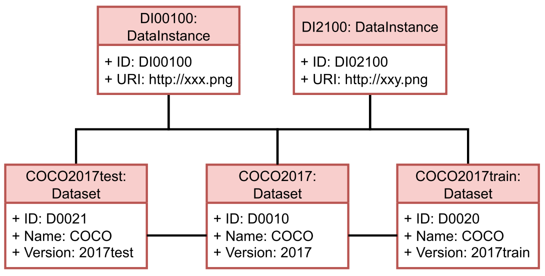


FIGURE 5. ***Dataset* model extract for a running example.**

C. CONFIGURATION PACKAGE

The Configuration package encompasses essential concepts that establish connections to various packages. These packages define the associated model, dataset, hardware specifications, predictions, etc. The Configuration model is related to multiple entities, ML Model, Hardware, Predication, and Dataset. It associates the model with dataset and hardware, indicating where the model is trained on with which dataset. A different associated training dataset will result in a different model, with different learned parameters/weights. Within the Configuration package, Hyperparameter model, Hardware model, and Configuration are three main components. These components collectively define the essential settings for both training and inference processes. They play a crucial role in determining the behavior and performance of the model throughout its lifecycle.

The Hardware model contains the concepts that denote the hardware that a model is trained on or executed on. The Hardware model comprises metadata that describes the hardware setting, associated with different HardwareAttributes. We list three types of hardware,

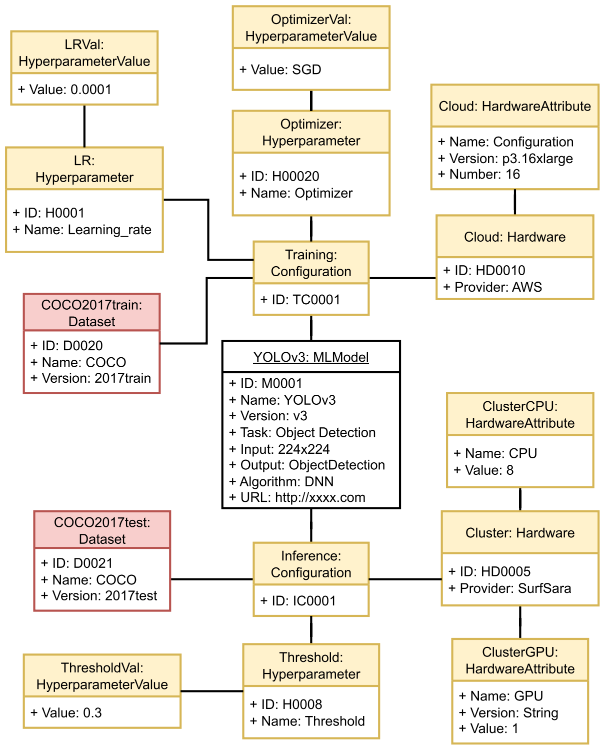


FIGURE 6. ***ML Model* and *Configuration* model extract for a running example.**

i.e., *Cloud* resources, *Edge* devices, and *Cluster*s. The *Cloud* resources associated with the public cloud services, such as AWS, Google Cloud, and Azure. An example of the hardware attributes for the *Cloud*, is the cloud configuration, e.g., 16 p3.16xlarge supported by AWS. For *Edge*, the hardware attributes include the type of the device (e.g., mobile phone, camera), storage capacity, memory, CPU, or GPU settings. Similar to *Edge*, a *Cluster* comprises metadata such as storage capacity, types, and the number of GPUs and CPUs.

The Configuration model is categorized into TrainingConfiguration and InferenceConfiguration. They have various sets of hyperparameters associated with different HyperparameterValue. For example, the former model specifies the hyperparameters related to optimization, e.g., type of optimizer, and learning rate, while these are not necessary for inference.

D. PREDICTION PACKAGE

The Prediction package contains the concepts that denote the model prediction/output, associated with the semantic concepts of the outputs (e.g., wheel to a car). The model elements of Prediction are presented with blue background.

The Prediction class allows the description of different types of tasks, i.e., regression and classification. Each Prediction is defined by *ID* and *Type* of the prediction. The prediction of a regression model is associated with a value. While the prediction of classification model can be further divided into multiple subclasses, e.g., binary classification, multiclass classification, and object detection. Classification model prediction is associated with a different number of classes. Specifically, ObjectDetection is associated with additional output, *BoundingBox*, which is defined by an array indicating the coordinate of the detected object in a picture. The classification prediction can be expanded and implemented for different kinds of models, e.g., image segmentation, with an additional attribute of an array.

Optionally a Class can be linked with a *Semantic Concept* from a knowledge base, thus allowing complex reasoning. Consider the example that an ML practitioner is building an ML model for image classification of cars, and she tries to conduct a model diagnosis. She may have several questions: what makes the model identify a car as a car? What are the semantic concepts that the model is capable of identifying? Are the wheels that make it believe that it is a car? To support such use-cases, the *metamodel* allows storing information about inference performance on specific data instances, which can be used to describe the behavior of a model, i.e., in what circumstances a model can perform well and why. Such information and awareness of the model prediction may significantly improve ML model interpretability in various applications such as health care, law enforcement, and finance.

FIGURE 4 presents an extract of the ML Model and Prediction as an example. *YOLOv3* and *YOLOv4* are two example models. Both models tackle object detection tasks, and the algorithm is *DNN*. Their output follows *ObjectDetection* types of prediction, with different classes (e.g., car and person in this case) and associated bounding boxes that locate the detected object. The object classes are related to different semantic concepts, for example, the wheel as part of the car, and the face as part of a person.

E. EVALUATION PACKAGE

The performance evaluation of an ML model is critical during both the training and deployment phases. The practitioners need to deploy a suitable ML model for the task given a specified environment, e.g., Hardware parameters. A mismatch of the deployment will lead to latency issues or reliability concerns, which results in user dissatisfaction. Hence, we present the Metric model to denote the model performance associated with the Configuration model. Thus the model performance is related to the model, hyper-parameter settings, hardware, and applied dataset. Specifically, we associate Metric model with TrainingConfiguration and InferenceConfiguration, since a model may train on a dataset while inference on another dataset. A Configuration is related to zero or more Metric, each associated with a unique MetricValue.

The Metric is categorized into multiple types of metrics, e.g., ExecutionMetric such as MemoryFootprint and Executiontime; and *Prediction Metric* denoting the correctness of the performance.

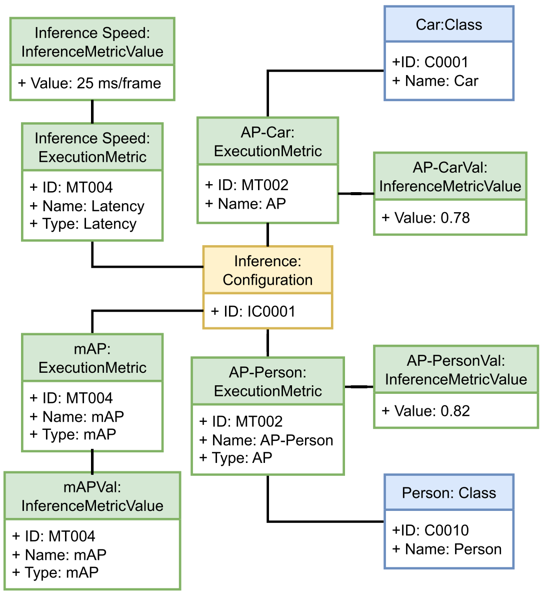


FIGURE 7. *Evaluation* model extract for a running example.

Evaluation Metric contains metric-related metadata. Different models will need different evaluation metrics, for instance, a regression model would have a *Mean Square Error - MSE*, while a multi-class classification model would have an *accuracy* value for each class.

FIGURE 7 presents the example of Evaluation package. The model inference configuration with ID *IC0001* is illustrated in FIGURE 6, associated with execution hardware, ML model, and dataset. To present the performance of a model under specified configuration settings, we include various types of metrics. In general, two kinds of metrics are introduced, i) correctness performance, verifying the prediction performance of the model, ii) and execution performance, including the runtime and memory footprint of the model. In the example, the latency and memory of the model are recorded, with a latency of 25ms and 90MB for the model size. In terms of correctness performance, we apply the common metrics to evaluate the object detection model. The metrics included in the example are average precision (AP) and mean average precision (mAP). In particular, we provide the AP metric for each class, allowing practitioners to assess the model's performance at a granular class level. Compared to the existing benchmarks or platforms that only report aggregate results, the performance metric that we include is more fine-grained.

V. MACARONI: A REFERENCE IMPLEMENTATION OF AN ADVANCED MODEL ZOO

This section describes the architecture and functionality of *Macaroni* [32], a tool designed and implemented to demonstrate how our metamodel can be used in ML model zoos. The implementation[4] is based on the metadata model described in the previous section. The architecture of Macaroni is shown

[4]https://sites.google.com/view/macaroni-model-zoo/home

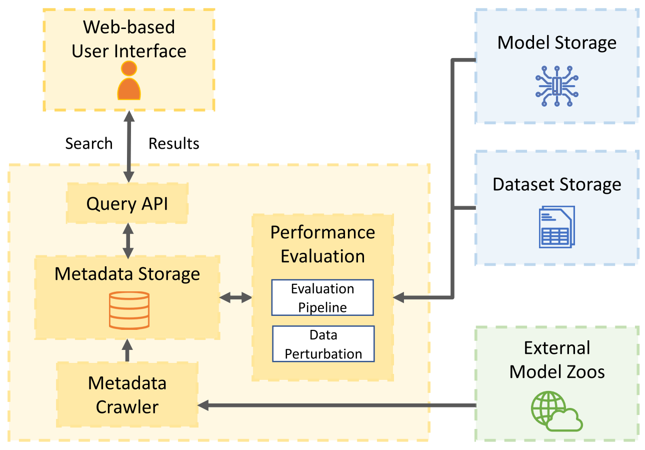


FIGURE 8. The structure of our metadata management tool.

in FIGURE 8. The tool is currently designed to retrieve metadata from different sources, integrate and enrich them, and allow for complex retrieval queries. The available APIs also allow for model execution and model composition.

A. ARCHITECTURE OVERVIEW

The system includes a web-based interface as a front-end to present the metadata and a back-end with storage and computation. The system comprises three main components: i) user interface; ii) acquiring and storing metadata; iii) storing and serving models and datasets. The second component is the core of the system. The system is built not only to present and query metadata but also to support ML model serving.

1) USER INTERFACE

Users can explore and query the metadata of the model zoo. We provide interfaces for users to filter and retrieve information. Some examples of the interfaces can be seen in FIGURE 9. The ingestion API allows users to ingest information regarding different artifacts. The metadata being retrieved is presented in an interactive visualization.

2) METADATA OBTAIN AND STORAGE

Recent works developed tools/systems to record metadata for the purpose of monitoring the experiments, especially during model training. Works such as Cerebro [73], MLflow [15], and ModelDB [33] provide APIs/logging libraries for users to track and log interested metadata in different levels of details. While others, such as ModelKB [35], automate the extraction process by identifying metadata from the source code of different deep learning frameworks.

In this work, we obtain the metadata in multiple ways and process and store it in the back-end. We collect metadata in the following three ways. i) A user can add the metadata of a model or a dataset by filling in specified fields, e.g., the content presented in a model card. Then such information is processed by the *Ingestion API* and converted into structured representation according to the above-mentioned metadata model, and stored in the *Metadata Storage*. ii) The tool can also automatically extract information from external model zoos, e.g., HuggingFace and PyTorch Hub. We can extract metadata from their API or information from the web pages and record the source of the external metadata in our metadata field. iii) To gather the information regarding the model performance, we apply a third way to obtain the metadata. We obtain the performance by evaluating the model on a dataset offline with specified hardware settings. This process utilizes cloud resources and is performed offline.

The metadata is stored in the structure described by the data model (Section IV). We implemented the data model and stored the metadata in MongoDB.

3) MODEL/DATA STORAGE AND SERVING

To support performance evaluation, we store related models and data scripts/files. The models and data are further applied with an automated evaluation pipeline (in the following subsection V-B). The models are perceived in docker images, and they also support model serving. Users can apply the model to their data and obtain prediction results.

B. PERFORMANCE EVALUATION AND AUTOMATED PIPELINE

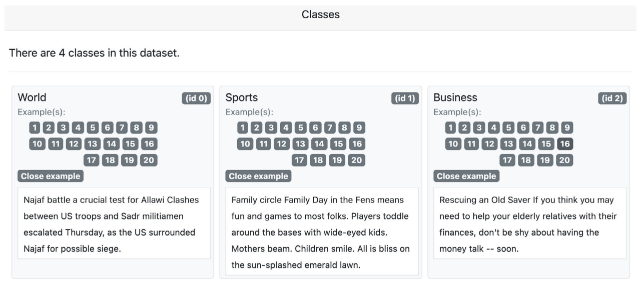
The tool supports the integration of models described and hosted in external model zoos. By the time we were writing the paper, we had imported more than 171k models hosted on the HuggingFace and FiftyOne model zoos, among which 986 of them were evaluated on 14 different datasets in texts or images. We develop an automated pipeline to execute/evaluate models from various platforms adapted from external APIs.

The performance metadata is gathered in three steps. i) The model is evaluated using the API from the model zoo the model was extracted from. ii) The output from the inference is then processed and transformed to a standardized format. iii) Finally, the performance of the model is evaluated based on the predictions, and the values will be stored in the proposed structure.

It is important to note that external APIs may have their own methods for performance calculation that deviate from the norm, or may not have some capability at all (performance of a class). To ensure the comparability of model performance, we implement unified evaluation methods.

C. RETRIEVAL AND EXPLORATION OF MODEL REPOSITORY

Throughout the ML lifecycle, ML practitioners will require different metadata for tracking the ML model status, editing, comparing, or reporting. An ML practitioner often needs to query models in large repositories with complex filtering conditions, e.g., data instance, performance, and inherit mechanism. In TABLE 2, we list a few example queries revealing different properties of the metadata. For example, Queries 1 and 2 require the metadata regarding the dataset, i.e., its attribute and source. This type of metadata helps practitioners understand the dataset, which allows them to


Classes

There are 4 classes in this dataset.

World (id 0) — Example(s): 1–20 — Close example

Najaf battle a crucial test for Allawi Clashes between US troops and Sadr militiamen escalated Thursday, as the US surrounded Najaf for possible siege.

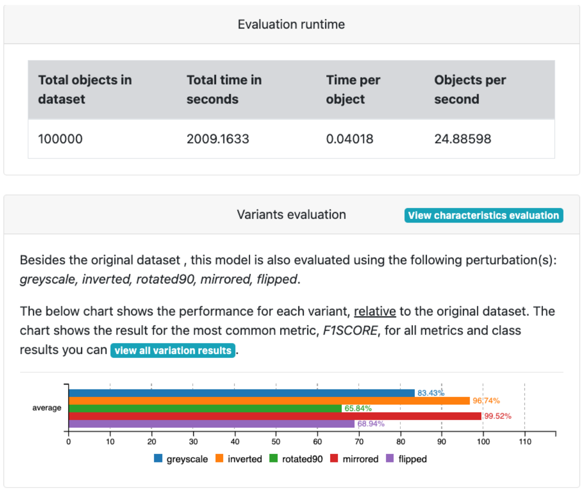
Sports (id 1) — Example(s): 1–20 — Close example

Family circle Family Day in the Fens means fun and games to most folks. Players toddle around the bases with wide-eyed kids. Mothers beam. Children smile. All is bliss on the sun-splashed emerald lawn.

Business (id 2) — Example(s): 1–20 — Close example

Rescuing an Old Saver If you think you may need to help your elderly relatives with their finances, don't be shy about having the money talk -- soon.

(a) Dataset page with data instance examples


Evaluation runtime

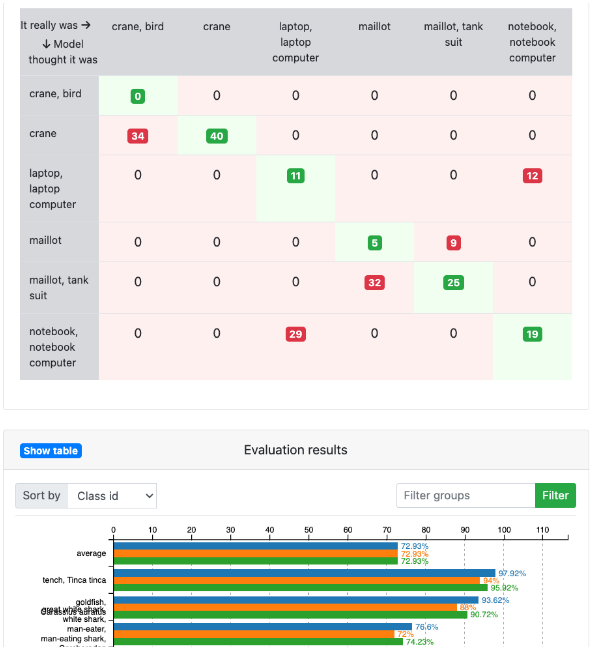
Total objects in dataset	Total time in seconds	Time per object	Objects per second
100000	2009.1633	0.04018	24.88598

Variants evaluation — View characteristics evaluation

Besides the original dataset , this model is also evaluated using the following perturbation(s): *greyscale, inverted, rotated90, mirrored, flipped.*

The below chart shows the performance for each variant, relative to the original dataset. The chart shows the result for the most common metric, *F1SCORE*, for all metrics and class results you can view all variation results.

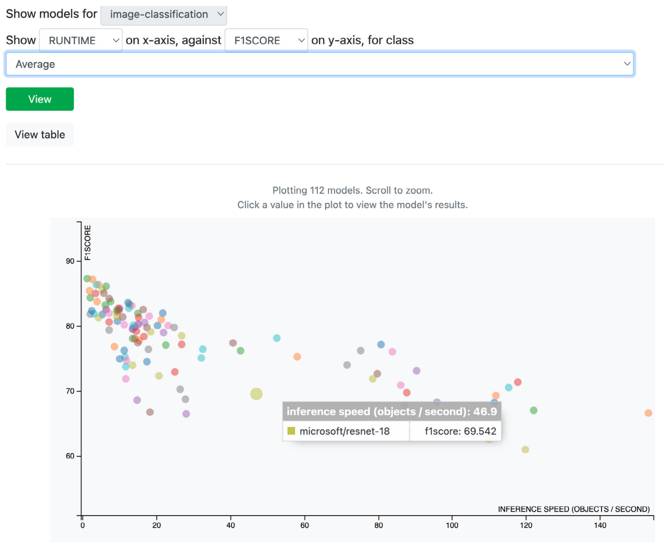
(b) Model page with aggregated performance


It really was → ↓ Model thought it was	crane, bird	crane	laptop, laptop computer	maillot	maillot, tank suit	notebook, notebook computer
crane, bird	0	0	0	0	0	0
crane	34	40	0	0	0	0
laptop, laptop computer	0	0	11	0	0	12
maillot	0	0	0	5	9	0
maillot, tank suit	0	0	0	32	25	0
notebook, notebook computer	0	0	29	0	0	19

Show table — Evaluation results

Sort by Class id — Filter groups — Filter

(c) Model page with fine-grained performance


Show models for image-classification

Show RUNTIME on x-axis, against F1SCORE on y-axis, for class Average

View

View table

Plotting 112 models. Scroll to zoom.
Click a value in the plot to view the model's results.

F1SCORE
90
80
70
60
inference speed (objects / second): 46.9
microsoft/resnet-18
f1score: 69.542
INFERENCE SPEED (OBJECTS / SECOND)
0
20
40
60
80
100
120
140

(d) Model comparison page

FIGURE 9. **The interfaces of Macaroni, including the dataset page, model page, and model comparison.**

gain insights into the characteristics and properties of the data and thus determine the relevant features in the feature selection and engineering stage. Queries such as Queries 3 to 6 require other metadata properties, such as the model performance and its interpretability. These properties are crucial for model discovery and comparison, and in addition, assist in decision-making processes and solving complex analytic problems. Query 7, on the other hand, requires more complex information regarding the inference performance with specified hardware settings. For example, an edge device may have constraints such as limited computation power and storage. To answer this query, practitioners will need to obtain the model performance of different objectives, e.g., inference speed and memory footprint.

The interface of the tool aims to let users find a model that is relevant to them. To do this, users can filter all available models by relevant properties, e.g., the name, task, or training dataset of the model. Once a model is selected, the user is presented with an overview of all available metadata, alongside the average evaluation results of the model (such as inference time and accuracy, if available) and a brief description of the associated dataset (if available).

If evaluation results are available, the user can opt to view more detailed information. We present the model performance with the following visualization types.

- *Table*. The tables present the (aggregated/disaggregated) performance results of a model with numbers, such that

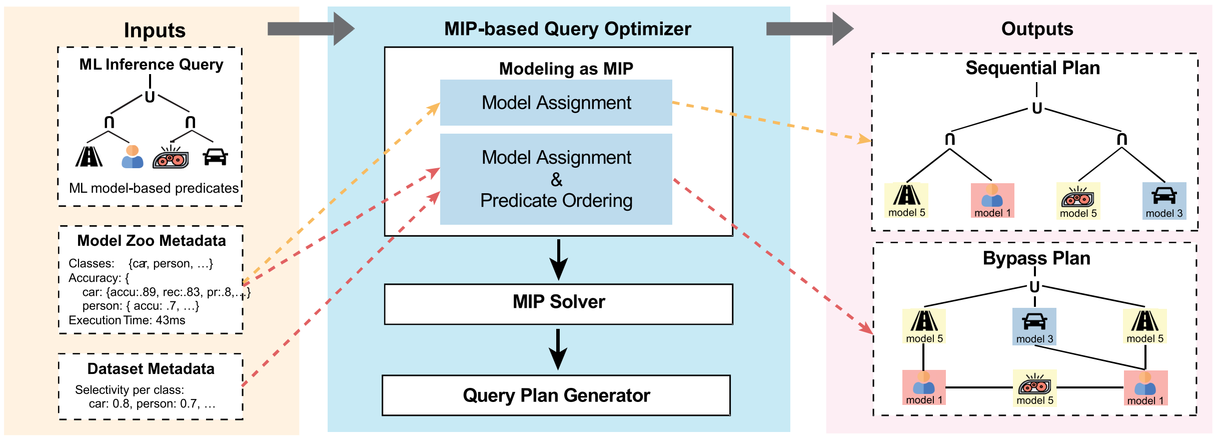


FIGURE 10. **ML inference query optimization.**

users can identify the best score of model performance on each task or on average.

- *Bar chart*. Bar charts and tables are interchangeable when presenting the performance of a model. For a clearer presentation, we only apply bar charts when comparing models. Users can also select the interested evaluation metrics and tasks for presentation.
- *Confusion matrix*. A confusion matrix is useful for model explanation, as users can observe when the model performs poorly. We also record the predictions of the model on data instances. Users can further explore the performance of the model by investigating the wrong predictions given the data examples.
- *Scatter plot*. As earlier stated, accuracy shall not be the only evaluation criterion of model performance, especially given complex requirements in the production environment. Specially, we also support comparing models on multiple objectives, e.g., accuracy against inference speed. For instance, one may observe that no single model can dominate in all objectives, e.g., having the highest accuracy score while being the most efficient to run.

D. USAGE AND FUNCTIONALITY

The querying of the metadata is performed in an online manner, while obtaining and updating the metadata can be processed offline. For instance, the metadata can be extended by evaluating the model and crawling the external model zoos regularly, e.g., once a week. Users can also trigger to evaluate the model and push the results to the metadata storage.

The tool supports different evaluation metrics, accuracy, inference speed, memory footprint, etc. Specifically, the accuracy of a model is not only presented in aggregated results of a task but also at a class level. Besides ingesting, extracting, and storing the metadata, our proposed tool allows a user to i) retrieve the models that help them identify a model, ii) compare multiple models, iii) or explore the properties of models/data by composing queries on the metadata.

VI. APPLICATION: MODEL COMPOSITION UNDER CONSTRAINTS

Now we introduce a more advanced yet common use case. With metadata being captured and well-represented, ML practitioners can make good use of the models trained offline and apply them to answer complex, ad-hoc inference queries.

Recent research has focused on ML applications for different modalities. For example, systems have been built to serve ML models for specific tasks [74]. Others aim to accelerate ML inference on domain tasks, such as NoScope [74], and PP [75]. These applications have shown a trend of applying model composition (usually with multiple models in cascades or in sequence) for complex tasks. The key idea of these works is to filter out insignificant data as early as possible, which is extremely useful when processing large-scale data, e.g., video, or streaming data, e.g., tweets.

The trend of applying and optimizing the usage of ML models in complex tasks has provided insights on how to manage ML models to better serve the tasks. A solution is to identify the capability of the models as fine-grained as possible. One of our previous works [69] has proposed to optimize for ML inference query by utilizing the metadata of ML models, especially the performance of models in multiple objectives.

As shown in FIGURE 10, the ad-hoc query can consist of multiple ML inference tasks with different dependencies and relations. Moreover, the query can be composed of specified constraints/requirements (e.g., latency and accuracy restrictions). Practitioners can select a composition of models from the model zoo to answer the ad-hoc queries. Optimization can be applied to further increase the efficiency of answering the query by carefully scheduling the tasks in a different order and applying early filtering, as the Bypass Plan in FIGURE 10. With the same example in FIGURE 10, an ML practitioner would like to design an application for video analysis that can capture a pedestrian crossing the road or the rear light of the car in front getting red. Since the data volume is significant and latency is also an essential factor to consider, the practitioner should select image classification or object detection models with fast inference speed. And the inference speed is greatly affected by the hardware being applied. If the application is deployed on a mobile phone, then the memory footprint is also a fundamental objective to be concerned with. To identify which set of models could best address the query and constraints, they may require information regarding the model performance with different objectives (e.g., accuracy, inference speed, memory footprint). The metadata of the dataset can also provide information to detect concept drift, for instance.

VII. CONCLUSION AND OUTLOOK

In this paper, we advocate for the need for a structured, queryable, and comprehensive metadata representation for model zoos. We propose a metadata model for such metadata representation to tackle different use cases. We also develop a tool that helps practitioners to manage and query the metadata.

We urge practitioners to prioritize the collection and organization of metadata, utilizing it for future applications. In order to enhance the applications of metadata, we present several aspects that can be explored in future research endeavors.

A. INTEGRATION OF ML MODEL METADATA TO CURRENT PLATFORMS

Existing work has developed tools to record (log/extract automatically) metadata during the ML lifecycle. Recent works only identify the public pre-trained model. Future work can integrate the systems seamlessly such that practitioners can have access to self-trained models as well as public pre-trained models.

B. NLP-BASED EXTRACTION OF METADATA FROM TEXT

With the support of large language models, future research can develop tools to extract useful information from the textual descriptions in the model and data cards by applying natural language processing techniques and mapping it to the predefined metadata representation.

C. PERSONALIZED AI AND FINETUNING

Many applications, such as behavior detection and virtual assistants, are user-specific and take into account user behaviors and preferences. Companies adapt the models to their own datasets and context by re-training and finetuning the models. Instead of randomly searching the hyper-parameter values to train a model from scratch, practitioners can utilize the metadata to accelerate the learning process by setting a warm-start for hyper-parameter search [21]. These AI applications can utilize the capabilities of the pre-trained models that already have a good performance on a certain task. We support these use cases by providing rich and comprehensive metadata, either the configuration settings or performance evaluation.

D. PROVENANCE, LINEAGE, AND VERSIONING

Researchers also propose to utilize metadata for other purposes, such as workflow data provenance [76], [77] and ML pipeline lineage [30], [78], [79]. ModelHub [34] and Modeldb [33], on the other hand, track metadata about models throughout the lifecycle and provide version management. However, they focus on the abstractions of the model, while they lack information on the model performance under different hardware settings (e.g., inference speed, accuracy, memory footprint).

REFERENCES

[1] Z. Yang, Z. Wang, Y. Huang, Y. Lu, C. Li, and X. S. Wang, "Optimizing machine learning inference queries with correlative proxy models," *Proc. VLDB Endowment*, vol. 15, no. 10, pp. 2032–2044, Jun. 2022.
[2] F. Romero, J. Hauswald, A. Partap, D. Kang, M. Zaharia, and C. Kozyrakis, "Optimizing video analytics with declarative model relationships," *Proc. VLDB Endowment*, vol. 16, no. 3, pp. 447–460, Nov. 2022.
[3] C. Hogan and G. Sistu, "Automatic vehicle ego body extraction for reducing false detections in automated driving applications," in *Proc. Irish Conf. Artif. Intell. Cogn. Sci.* Cham, Switzerland: Springer, 2022, pp. 264–275.
[4] T. Gillespie, "Content moderation, AI, and the question of scale," *Big Data Soc.*, vol. 7, no. 2, Jul. 2020, Art. no. 205395172094323.
[5] D. Nallaperuma, R. Nawaratne, T. Bandaragoda, A. Adikari, S. Nguyen, T. Kempitiya, D. De Silva, D. Alahakoon, and D. Pothuhera, "Online incremental machine learning platform for big data-driven smart traffic management," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4679–4690, Dec. 2019.
[6] F. L. Sánchez, I. Hupont, S. Tabik, and F. Herrera, "Revisiting crowd behaviour analysis through deep learning: Taxonomy, anomaly detection, crowd emotions, datasets, opportunities and prospects," *Inf. Fusion*, vol. 64, pp. 318–335, Dec. 2020.
[7] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3296–3297.
[8] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," in *Proc. Conf. Fairness, Accountability, Transparency*, Jan. 2019, pp. 220–229.
[9] S. M. Jain, "Hugging face," in *Introduction to Transformers for NLP*. Cham, Switzerland: Springer, 2022, pp. 51–67.
[10] *Pytorch Hub*. Accessed: Sep. 10, 2023. [Online]. Available: https://pytorch.org/hub/
[11] *Tensorflow Hub*. Accessed: Sep. 10, 2023. [Online]. Available: https://www.tensorflow.org/hub
[12] *Papers With Code*. Accessed: Sep. 10, 2023. [Online]. Available: https://paperswithcode.com/sota
[13] *Openvino Open Model Zoo*. Accessed: Sep. 10, 2023. [Online]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/master
[14] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "Dawnbench: An end-to-end deep learning benchmark and competition," *Training*, vol. 100, no. 101, p. 102, 2017.
[15] A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S. A. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, A. Singh, F. Xie, M. Zaharia, R. Zang, J. Zheng, and C. Zumar, "Developments in MLflow: A system to accelerate the machine learning lifecycle," in *Proc. 4th Int. Workshop Data Manage. End End Mach. Learn.*, Jun. 2020, pp. 1–4.
[16] *Weights & Biases, the AI Developer Platform*. Accessed: Sep. 10, 2023. [Online]. Available: https://wandb.ai/site
[17] *Clearml Auto-Magical Suite of Tools to Streamline Your ML Workflow*. Accessed: Sep. 10, 2023. [Online]. Available: https://clear.ml/docs/latest
[18] *Comat Less Friction, Mode ML*. Accessed: Aug. Sep. 10, 2023. [Online]. Available: https://www.comet.com/site/
[19] Z. Abedjan, L. Golab, and F. Naumann, "Profiling relational data: A survey," *VLDB J.*, vol. 24, no. 4, pp. 557–581, Aug. 2015.

[20] M. Yurrita, T. Draws, A. Balayn, D. Murray-Rust, N. Tintarev, and A. Bozzon, "Disentangling fairness perceptions in algorithmic decision-making: The effects of explanations, human oversight, and contestability," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, New York, NY, USA, Apr. 2023, pp. 1–21.

[21] S. Schelter, F. Biessmann, T. Januschowski, D. Salinas, S. Seufert, and G. Szarvas, "On challenges in machine learning model management," Amazon, Tech. Rep., 2018.

[22] M. Schlegel and K.-U. Sattler, "Management of machine learning lifecycle artifacts: A survey," *ACM SIGMOD Rec.*, vol. 51, no. 4, pp. 18–35, 2023.

[23] I. F. Ilyas and X. Chu, *Data Cleaning*. San Rafael, CA, USA: Morgan & Claypool, 2019.

[24] C. Yang, Y. Akimoto, D. W. Kim, and M. Udell, "OBOE: Collaborative filtering for AutoML model selection," 2018, *arXiv:1808.03233*.

[25] J. Klaise, A. Van Looveren, G. Vacanti, and A. Coca, "Alibi explain: Algorithms for explaining machine learning models," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 8194–8200, 2021.

[26] P. Guo, B. Hu, and W. Hu, "Sommelier: Curating DNN models for the masses," in *Proc. Int. Conf. Manage. Data*, Jun. 2022, pp. 1876–1890.

[27] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (MLOps): Overview, definition, and architecture," *IEEE Access*, vol. 11, pp. 31866–31879, 2023.

[28] J. Talukdar, S. Gupta, P. S. Rajpura, and R. S. Hegde, "Transfer learning for object detection using state-of-the-art deep neural networks," in *Proc. 5th Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Feb. 2018, pp. 78–83.

[29] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[30] S. Schelter, J.-H. Boese, J. Kirschnick, T. Klein, and S. Seufert, "Automatically tracking metadata and provenance of machine learning experiments," in *Proc. Mach. Learn. Syst. Workshop (NIPS)*, 2017, pp. 27–29.

[31] P. Agrawal, R. Arya, A. Bindal, S. Bhatia, A. Gagneja, J. Godlewski, Y. Low, T. Muss, M. M. Paliwal, and S. Raman, "Form for machine learning," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1803–1816.

[32] Z. Li, H. Kant, R. Hai, A. Katsifodimos, and A. Bozzon, "Macaroni: Crawling and enriching metadata from public model zoos," in *Proc. Int. Conf. Web Eng.* Cham, Switzerland: Springer, 2023, pp. 376–380.

[33] M. Vartak, H. Subramanyam, W.-E. Lee, S. Viswanathan, S. Husnoo, S. Madden, and M. Zaharia, "MODELDB: A system for machine learning model management," in *Proc. Workshop Hum. Loop Data Anal.*, 2016, pp. 1–3.

[34] H. Miao, A. Li, L. S. Davis, and A. Deshpande, "Towards unified data and lifecycle management for deep learning," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 571–582.

[35] G. Gharibi, V. Walunj, R. Nekadi, R. Marri, and Y. Lee, "Automated end-to-end management of the modeling lifecycle in deep learning," *Empirical Softw. Eng.*, vol. 26, no. 2, pp. 1–33, Mar. 2021.

[36] J. Tsay, T. Mummert, N. Bobroff, A. Braz, P. Westerink, and M. Hirzel, "Runway: Machine learning model experiment management tool," in *Proc. Conf. Syst. Mach. Learn. (SysML)*, 2018, pp. 1–3.

[37] *Ml Metadata TFX Tensorflow*. Accessed: Sep. 10, 2023. [Online]. Available: https://www.tensorflow.org/tfx/guide/mlmd

[38] E. Brumbaugh, "Bighead: A framework-agnostic, end-to-end machine learning platform," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2019, pp. 551–560.

[39] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A \$$Low-Latency$\$ online prediction serving system," in *Proc. 14th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 613–627.

[40] P. Kraft, D. Kang, D. Narayanan, S. Palkar, P. Bailis, and M. Zaharia, "Willump: A statistically-aware end-to-end optimizer for machine learning inference," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 147–159, Mar. 2020.

[41] A. Balasubramanian, A. Kumar, Y. Liu, H. Cao, S. Venkataraman, and A. Akella, "Accelerating deep learning inference via learned caches," 2021, *arXiv:2101.07344*.

[42] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "PMLB: A large benchmark suite for machine learning evaluation and comparison," *BioData Mining*, vol. 10, no. 1, pp. 1–13, Dec. 2017.

[43] R. S. Olson, "A system for accessible artificial intelligence," in *Genetic Programming Theory and Practice XV*. Cham, Switzerland: Springer, 2018, pp. 121–134.

[44] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "MoleculeNet: A benchmark for molecular machine learning," *Chem. Sci.*, vol. 9, no. 2, pp. 513–530, 2018.

[45] Y. Alaudah, P. Michałowicz, M. Alfarraj, and G. AlRegib, "A machine-learning benchmark for facies classification," *Interpretation*, vol. 7, no. 3, pp. SE175–SE187, Aug. 2019.

[46] R. Chard, Z. Li, K. Chard, L. Ward, Y. Babuji, A. Woodard, S. Tuecke, B. Blaiszik, M. J. Franklin, and I. Foster, "DLHub: Model and data serving for science," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2019, pp. 283–292.

[47] Ž. Avsec, R. Kreuzhuber, J. Israeli, N. Xu, J. Cheng, A. Shrikumar, A. Banerjee, D. S. Kim, L. Urban, and A. Kundaje, "Kipoi: Accelerating the community exchange and reuse of predictive models for genomics," *BioRxiv*, 2018.

[48] J. Giner-Miguelez, A. Gómez, and J. Cabot, "DescribeML: A tool for describing machine learning datasets," in *Proc. 25th Int. Conf. Model Driven Eng. Lang. Syst., Companion*, Oct. 2022, pp. 22–26.

[49] R. Hai, C. Koutras, A. Ionescu, Z. Li, W. Sun, J. van Schijndel, Y. Kang, and A. Katsifodimos, "Amalur: Data integration meets machine learning," in *Proc. IEEE 39th Int. Conf. Data Eng. (ICDE)*, Apr. 2023, pp. 3729–3739.

[50] I. Foster, J. Vockler, M. Wilde, and Y. Zhao, "Chimera: A virtual data system for representing, querying, and automating data derivation," in *Proc. 14th Int. Conf. Sci. Stat. Database Manage.*, Jul. 2002, pp. 37–46.

[51] Y. Zhang, F. Xu, E. Frise, S. Wu, B. Yu, and W. Xu, "DataLab: A version data management and analytics system," in *Proc. IEEE/ACM 2nd Int. Workshop Big Data Softw. Eng. (BIGDSE)*, May 2016, pp. 12–18.

[52] A. Gandhi, Y. Asada, V. Fu, A. Gemawat, L. Zhang, R. Sen, C. Curino, J. Camacho-Rodríguez, and M. Interlandi, "The tensor data platform: Towards an ai-centric database system," 2022, *arXiv:2211.02753*.

[53] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. Iii, and K. Crawford, "Datasheets for datasets," *Commun. ACM*, vol. 64, no. 12, pp. 86–92, Dec. 2021.

[54] M. Miceli, T. Yang, L. Naudts, M. Schuessler, D. Serbanescu, and A. Hanna, "Documenting computer vision datasets: An invitation to reflexive data practices," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Mar. 2021, pp. 161–172.

[55] K. L. Boyd, "Datasheets for datasets help ML engineers notice and understand ethical issues in training data," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, pp. 1–27, Oct. 2021.

[56] P. Mattson, C. Cheng, G. Diamos, C. Coleman, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, and V. Bittorf, "MLPerf training benchmark," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 336–349, Mar. 2020.

[57] V. J. Reddi, "MLPerf inference benchmark," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, May 2020, pp. 446–459.

[58] R. Adolf, S. Rama, B. Reagen, G.-Y. Wei, and D. Brooks, "Fathom: Reference workloads for modern deep learning methods," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Sep. 2016, pp. 1–10.

[59] L. J. Miranda. (2021). *Towards Data-Centric Machine Learning: A Short Review*. [Online]. Available: ljvmiranda921.github.io

[60] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey and benchmarking of machine learning accelerators," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2019, pp. 1–9.

[61] A. Svyatkovskiy, J. Kates-Harbeck, and W. Tang, "Training distributed deep recurrent neural networks with mixed precision on GPU clusters," in *Proc. Mach. Learn. HPC Environ.*, Nov. 2017, pp. 1–8.

[62] A. Balayn, P. Soilis, C. Lofi, J. Yang, and A. Bozzon, "What do you mean? Interpreting image classification with crowdsourced concept extraction and analysis," in *Proc. Web Conf.*, Apr. 2021, pp. 1937–1948.

[63] C. Coleman, D. Kang, D. Narayanan, L. Nardi, T. Zhao, J. Zhang, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "Analysis of DAWNBench, a time-to-accuracy machine learning performance benchmark," *ACM SIGOPS Operating Syst. Rev.*, vol. 53, no. 1, pp. 14–25, Jul. 2019.

[64] R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the machine learning lifecycle: Desiderata, methods, and challenges," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–39, Jun. 2022.

[65] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data lifecycle challenges in production machine learning: A survey," *ACM SIGMOD Rec.*, vol. 47, no. 2, pp. 17–28, Dec. 2018.

[66] C. Chai, J. Wang, Y. Luo, Z. Niu, and G. Li, "Data management for machine learning: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4646–4667, May 2023.

[67] A. Balayn, C. Lofi, and G.-J. Houben, "Managing bias and unfairness in data for decision support: A survey of machine learning and data engineering approaches to identify and mitigate bias and unfairness within data management and analytics systems," *VLDB J.*, vol. 30, no. 5, pp. 739–768, Sep. 2021.
[68] T. Wolf, "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*, 2020, pp. 38–45.
[69] Z. Li, M. Schönfeld, W. Sun, M. Fragkoulis, R. Hai, A. Bozzon, and A. Katsifodimos, "Optimizing ML inference queries under constraints," in *Proc. Int. Conf. Web Eng.* Cham, Switzerland: Springer, 2023, pp. 51–66.
[70] Y. Wu, M. Lentz, D. Zhuo, and Y. Lu, "Serving and optimizing machine learning workflows on heterogeneous infrastructures," 2022, *arXiv:2205.04713*.
[71] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, Tech. Rep., 2018.
[72] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[73] A. Kumar, S. Nakandala, Y. Zhang, S. Li, A. Gemawat, and K. Nagrecha, "CEREBRO: A layered data platform for scalable deep learning," in *Proc. CIDR*, 2021, pp. 1–10.
[74] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "NoScope: Optimizing neural network queries over video at scale," 2017, *arXiv:1703.02529*.
[75] Y. Lu, A. Chowdhery, S. Kandula, and S. Chaudhuri, "Accelerating machine learning inference with probabilistic predicates," in *Proc. Int. Conf. Manage. Data*, May 2018, pp. 1493–1508.
[76] H. Miao, A. Chavan, and A. Deshpande, "ProvDB: Lifecycle management of collaborative analysis workflows," in *Proc. 2nd Workshop Hum. Loop Data Anal.*, May 2017, pp. 1–6.
[77] Z. Zhang, E. R. Sparks, and M. J. Franklin, "Diagnosing machine learning pipelines with fine-grained lineage," in *Proc. 26th Int. Symp. High-Perform. Parallel Distrib. Comput.*, Jun. 2017, pp. 143–153.
[78] S. Grafberger, S. Guha, J. Stoyanovich, and S. Schelter, "MLINSPECT: A data distribution debugger for machine learning pipelines," in *Proc. Int. Conf. Manage. Data*, Jun. 2021, pp. 2736–2739.
[79] T. van der Weide, D. Papadopoulos, O. Smirnov, M. Zielinski, and T. van Kasteren, "Versioning for end-to-end machine learning pipelines," in *Proc. 1st Workshop Data Manage. End End Mach. Learn.*, May 2017, pp. 1–9.

ZIYU LI (Member, IEEE) received the B.S. degree from the South China University of Technology, China, in 2017, and the M.S. degree in computer science from the Delft University of Technology, The Netherlands, in 2019, where she is currently pursuing the Ph.D. degree with the Department of Software Technology. Her research interests include data management, machine learning, and metadata management.

HENK KANT obtained the M.S. and B.S. degrees from the Delft University of Technology, the Netherlands, in 2013 and 2023. The M.S. degree was with a specialization in Information Architecture.

RIHAN HAI (Member, IEEE) received the Ph.D. degree from RWTH Aachen University, Germany. She is currently an Assistant Professor with the Web Information Systems Group, Delft University of Technology, The Netherlands. Her research interests include data lakes, data integration, and data management for machine learning. She has served as a Program Committee Member of database conferences, such as VLDB, ICDE, and EDBT, and a Reviewer for journals, such as IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *The VLDBJ Journal*, and *JMLR*.

ASTERIOS KATSIFODIMOS received the Ph.D. degree from INRIA Saclay & University Paris 11. He is currently an Associate Professor with the Delft University of Technology, and a Visiting Academic with Amazon Web Services (AWS)—AI. Before that, he was with the SAP Innovation Center, Berlin, and Technical University (TU), Berlin. His research spans the areas of parallel data processing and Cloud computing, and data integration. His research on fault tolerance, aggregation methods and benchmarking has influenced the design of opensource stream processing engines, while his research group develops and maintains the dataset discovery system Valentine. He has received the ACM SIGMOD Research Highlights Award, in 2016, the EDBT Best Paper, in 2019, the Best Demo Award, in 2023, and the ACM SIGMOD Systems Award 2023. He is the instructor of the online MOOC "Taming Massive Data Streams" and is invited regularly to give talks at industry and research venues. He serves as an Associate Editor or a Program Committee Member for the data management conferences, such as VLDB, ICDE, SIGMOD, and EDBT.

MARCO BRAMBILLA (Member, IEEE) received the Ph.D. degree from Politecnico di Milano, in 2005. He is currently a Full Professor with Politecnico di Milano. His research interests include data science, software modeling languages, crowdsourcing, social media analysis, data-driven innovation, and big data architectures. He has been a Visiting Researcher with CISCO and UCSD, USA, and a Visiting Professor with Dauphine University, Paris. He is the main author of the OMG Standard Interaction Flow Modeling Language (IFML). He founded three startups and authored over 250 papers and two patents. He has coauthored five books on model-driven software engineering. He is an Associate Editor of the journals, such as *Web Engineering*, *Digital*, and *Advances in Human-Computer Interactions*. He is a member of the Steering Committee of the International Conference of Web Engineering.

ALESSANDRO BOZZON received the Ph.D. degree from Politecnico di Milano, in 2009. His Ph.D. thesis focused on model driven approaches for the design, development and automatic code generation of search based applications. He is currently a Professor in human-centered artificial intelligence and the Head of the Department of Sustainable Design Engineering, Delft University of Technology. He is a Principal Investigator of Urban Data and Intelligence with the Amsterdam Institute for Advanced Metropolitan Solutions (AMS), a member of the Steering Committee of the International Conference of Web Engineering (ICWE), and a member of the Steering Committee of the Human Computation and Crowdsourcing (HCOMP) Conference. His research interests include the intersection of human–computer interaction and machine learning. He has coauthored more than 200 papers in leading peer-reviewed international journals and conferences, where he also regularly serves as a senior program committee member.