# DRAFT PROPOSAL Medical Event Data Standard: An ML-oriented Interface for Medical Record Datasets

**Bert Arnrich**
Digital Health
Hasso Plattner Institute
bert.arnrich@hpi.de

**Edward Choi**
Kim Jaechul Graduate School of AI
KAIST
edwardchoi@kaist.ac.kr

**Jason A. Fries**
Center for Biomedical Informatics Research
Stanford University
jfries@stanford.edu

**Matthew B. A. McDermott**
Department of Biomedical Informatics
Harvard Medical School
matthew_mcdermott@hms.harvard.edu

**Jungwoo Oh**
Kim Jaechul Graduate School of AI
KAIST
ojw0123@kaist.ac.kr

**Tom J Pollard**
Institute for Medical Engineering and Science
Massachusetts Institute of Technology
tpollard@mit.edu

**Nigam Shah**
Center for Biomedical Informatics Research
Clinical Excellence Research Center
Stanford University
Technology and Digital Solutions
Stanford Healthcare
nigam@stanford.edu

**Ethan Steinberg**
Department of Computer Science
Stanford University
ethanid@stanford.edu

**Michael Wornow**
Department of Computer Science
Stanford University
mwornow@stanford.edu

**Robin van de Water**
Digital Health
Hasso Plattner Institute
robin.vandewater@hpi.de

## Abstract

We introduce the *Medical Event Data Standard* (MEDS), a simple dataset schema for machine learning over electronic health record (EHR) data. Unlike existing tools, pipelines, and common data models, MEDS is a minimal standard designed for maximum interoperability across datasets, existing tools, and model architectures. By providing a simple standardization layer between datasets and model-specific code, MEDS will enable more reproducible, robust, computationally performant, and collaborative machine learning research for EHR data. Alongside this report, we highlight several existing MEDS integrations with models, datasets, and tools, and will work actively with the community for further development and adoption.

Preprint. Under review.

## 1 Introduction

In many successful sub-fields of machine learning (ML), such as computer vision (CV) or natural language processing (NLP), the input data modality has a standard "input format" which is used widely across machine learning applications. For example, images for CV use a small set of widely recognized image encodings (*e.g.* JPG, PNG, etc.) while NLP uses strings for text.

These implicit standards yield a number of benefits, including the ability to naturally share model code that can work across diverse datasets of the appropriate modality; the ability to curate large, widely used benchmarks (*e.g.*, ImageNet [3], GLUE [20]); and the ability to organically develop shared, composable pipeline components that can be leveraged by the community to accelerate research over larger data scales (*e.g.*, torchvision transformations for data augmentation in CV [11] or Hugging Face Dataset transformations for NLP pipelines [21]).

Unfortunately, an ML-friendly data standard does not exist for medical record data. Current solutions such as common data models (CDMs) (e.g. OMOP-CDM [1]), interoperability formats (e.g. FHIR [2]), and end-to-end data preprocessing pipelines (e.g. PyHealth [23]) are not well-suited for modern deep learning development. While OMOP-CDM and other health CDMs are well optimized for federated observational statistical analyses and traditional ML models, but *pose challenges* for deep learning developers. In particular, the complexity of these CDMs (OMOP-CDM contains 394 fields across 39 tables) and the risk of inducing bias when transforming data to the requisite format present significant barriers for developing high-throughput ML systems that researchers must overcome with customized extraction and batching pipelines. This contributes to the reproducibility crisis in ML for healthcare [13] and limits the accessibility of the field to researchers without a clinical background. In contrast, while end-to-end pipelines such as PyHealth [23], TemporAI [15], ESGPT [12], and YAIB [19] offer simpler interfaces for non-clinical researchers; however, these pipelines operate in closed ecosystems, with limited options for transferring models across pipelines, datasets, and tasks.

In this proposal, we address this critical gap by introducing the *Medical Event Data Standard* (MEDS), a dataset schema for machine learning (ML) modeling over electronic health record (EHR) data that solves 3 critical problems:

1. MEDS enables researchers to write modeling code that works across datasets, thereby improving reproducibility and the ability to validate models across multiple datasets.
2. MEDS enables researchers to share data preprocessing code, improving productivity by allowing the community to easily build on the work of others.
3. MEDS enables health systems to easily take advantage of existing state-of-the-art health ML research by converting their data to the MEDS format. Due to its minimal, generalizable design, MEDS extraction ETLs are much simpler than traditional CDM ETLs and accordingly are lower-cost and present a reduced risk of data bias. MEDS datasets can also be derived from existing data models such as OMOP-CDM via standardized tools.

While existing CDMs such as OMOP-CDM are optimized for federated observational statistical analyses using classical, low-capacity models, MEDS is designed to simplify machine learning model development by minimizing dataset complexity and risk of transformation bias. MEDS datasets can be created via simple data extraction pipelines, and MEDS is expressly designed to empower an open, shared ecosystem of model and data pipeline components. MEDS does not provide pre-built models, but instead any user-developed models or model pipeline components that leverage MEDS-compliant inputs will be reuseable across other compliant datasets and in future pipelines. In particular, by leveraging the Hugging Face datasets library and ecosystem, MEDS data pre-processing and preliminary model components can be implemented in the form of simple dataset transformation functions that operate at a per-patient level. These transforms can then be natively applied to MEDS datasets through the Hugging Face library to translate from input data representations to model-specific, cacheable datasets that permit efficient batching. This capability will encourage the community to develop shareable, re-usable model components in a way that closed ecosystem end-to-end solutions do not. See Figure 1 for a visual representation.

Ultimately, we hope that MEDS can serve to unify and empower ML for health research efforts across datasets and modeling tasks. MEDS will make it far easier for researchers to profile competing methods against new models, thereby establishing more robust and scaleable benchmarking systems

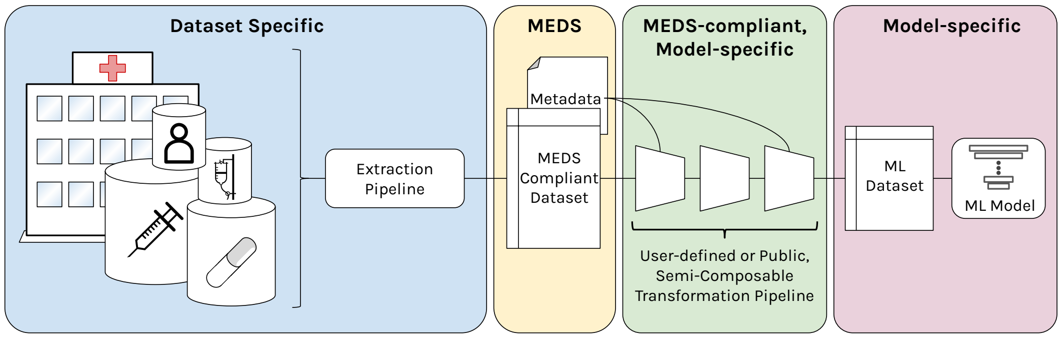

Figure 1: The Event Stream Data Standard (MEDS) defines a dataset and metadata schema such that, after any raw EHR dataset is extracted into an MEDS-compliant form, users can build streamlined, shareable, reproducible custom modeling pipelines atop that dataset. These pipelines consist first of composable transformations to the raw dataset's format, ultimately resulting in a cache-able dataset that can be used with their specific ML model code.

for ML algorithms over EHR data. Through its support of iterative, composable pipeline transformation functions, MEDS will also help to establish a more vigorous ecosystem of EHR-centric model components. Finally, through the use of state-of-the-art technology platforms and strong design principles, MEDS will also help enable more efficient and effective model development, especially on larger datasets and with higher-capacity models.

In the rest of this work we will provide a detailed overview of the MEDS schema, including its key design principles, concrete dataset schemas, and examples of how it can be used in diverse ways. We will then offer pointers to existing ETLs for common datasets; integrations with tools including ESGPT [12], FEMR, GenHPF [6], and YAIB [19]; and published models, including the foundation model CLMBR [22]. Finally, we will close with a discussion of some limitations of MEDS and its future roadmap.

## 2 Overview of the MEDS Schema

MEDS-compliant datasets are organized into two distinct schemas: (1) an Apache Arrow `patient` schema that contains per-patient recorded information and (2) a JSON `dataset_metadata` schema that contains overall information about a dataset.

### 2.1 Patient Schema

In Figure 2, we show the Apache Arrow `patient` schema of an MEDS-compliant dataset. Each row in the `patient` schema represents a single patient and contains all information about that patient from the source EHR or claims dataset. That information consists of a `patient_id`, a unique identifier for each patient, and a list of timestamped `events`. Each event corresponds to an instance of time when information was recorded about a patient and in turn contains a list of `measurements` that provide that information. Each measurement consists of a categorical code[1], an optional value (of varying types), and optional metadata. Example information that might form a measurement includes the assignment of a diagnosis code, a laboratory test, a filled out flowsheet, or event a full text note.

See Figure 3 for a visual example of a patient in this schema.

The requirements for data within the `patient` schema are quite flexible. Every patient is required to have a unique identifier, each event is required to have a unique time within a patient, and each measurement is required to have a code. Codes do not have a required format, but it is highly recommended that they be in the form of of `"VOCABULARY_NAME/CODE_TEXT"` when the code comes from a standard vocabulary with an OHDSI defined vocabulary name [2]. For example, it is recommended to represent the ICD10CM code I50.9 as `"ICD10CM/I50.9"`

[1]Categorical codes are currently recorded as Apache Arrow string types to enable integration with HF datasets, but this will be changed soon.

[2]https://github.com/OHDSI/Vocabulary-v5.0/wiki/Standardized-Vocabularies

```python
import pyarrow as pa

# Each dataset has custom metadata structures.
# For this example, the metadata is a struct with a single field.
custom_metadata = pa.struct([
    ("table", pa.string())
])

measurement = pa.struct([
    # A string code that describes what the event is.
    ("code", pa.string()),

    # Optional values for the event.
    # Any number of these can be null or not-null.
    ("text_value", pa.string()),
    ("numeric_value", pa.float32()),
    ("datetime_value", pa.timestamp("us")),

    # Per-event metadata that has a unique schema per dataset
    ("metadata", custom_metadata),
])


event = pa.struct([
    # The time at which an event occurs
    ("time", pa.timestamp("us")),

    # Each event consists of a series of measurements
    ("measurements", pa.list_(measurement))
])

patient = pa.schema([
    # The unique identifier for each patient
    ("patient_id", pa.int64()),

    # The events for that patient
    # These must be ordered by time
    ("events", pa.list_(event)),
])

    return patient
```

Figure 2: The MEDS `patient` schema which stores patient information. Each row in the schema consists of all data for a single patient, consisting of a `patient_id` and a list of `events`. Each event contains a timestamp and a list of measurements that occurred at that event.

```
meds_dataset = ...
# This pretty print function does not actually exist.
pretty_print(meds_dataset[0], drop_nulls=True)
{
    'patient_id': 3,
    'events': [
        {
            "time": datetime(2010, 2, 3, 11, 52, 13),
            "measurements": [
                {"code": "ADMISSION"},
                {"code": "LOINC/...", "numeric_value": ...},
                {"code": "LOINC/...", "text_value": ...},
            ]
        }, {
            "time": datetime(2010, 2, 4, 6, 10, 0),
            "measurements": [
                {"code": "LOINC/...", "numeric_value": ...},
                {"code": "LOINC/...", "numeric_value": ...},
                {"code": "LOINC/...", "text_value": ...},
            ]
        }, {
            "time": datetime(2010, 2, 5, 15, 0, 0),
            "measurements": [
                {"code": "DISCHARGE"},
                {"code": "ICD10CM/..."},
                {"code": "ICD10CM/..."},
            ]
        },
        ...
    ],
}
```

Figure 3: An example `patient` data row from a MEDS-compliant dataset.

Measurements may also have additional modifiers outside the scope of a single code and value; however, modelers should be cautious in attempting to use such modifiers as they will neither have a consistent schema nor be guaranteed to be populated across all MEDS-compliant datasets. To include modifiers in a dataset, the data owner can simply store an arbitrary Arrow type within the metadata field of the measurements struct. In this way, the core `patient` schema (defining patient identifiers, events, and measurements) does not change, but new data can still be included in a type-safe manner. We hope that as MEDS is further used and validated, some common metadata fields will become more standardized across datasets and perhaps moved into the universal `patient` schema itself.

## 2.2 Dataset Metadata Schema

Users may want to leverage dataset specific information when working with an MEDS-compliant dataset. In order to support that, we specify a common format for that information with a `dataset_metadata` JSON schema, as shown in Figure 4. We have added a variety of commonly useful fields to this schema, including versioning information, string descriptions for dataset-specific non-standard codes, and mappings from dataset-specific non-standard codes to public ontologies.

As MEDS is adopted, we hope that common metadata column conventions will emerge and be adopted by dataset curators in general and, therefore, that some such information can be reliably used by modelers in general. However, we leave the emergence of this to the community through the organic usage of this schema.

```
# The dataset metadata schema.

# Code specific metadata
code_metadata_entry = {
    "type": "object",
    "properties": {
        # A text description of the code
        "description": {"type": "string"},

        # Mappings between a code and either higher level codes or codes
    within standard ontologies
        "parent_codes": {"type": "array", "items": {"type": "string"}},
    },
}

# This is a simple code string to metadata entry map
code_metadata = {
    "type": "object",
    "additionalProperties": code_metadata_entry,
}

# Overall dataset metadata
dataset_metadata = {
    "type": "object",
    "properties": {
        "dataset_name": {"type": "string"},
        "dataset_version": {"type": "string"},
        "etl_name": {"type": "string"},
        "etl_version": {"type": "string"},
        "code_metadata": code_metadata,
    },
}
```

Figure 4: The MEDS `dataset_metadata` schema which stores general dataset specific metadata.

## 3 Collaborating Tools in the MEDS Ecosystem

Several existing tools, models, and pipelines have already established MEDS-compliant versions of their tools. We list each here.

**Framework for Electronic Medical Records (FEMR)** FEMR [22] provides an MEDS-compliant suite of data preprocessing tools for building machine learning models on top of EHR and claims data at scale. FEMR has been used for the development of multiple clinical foundation models (MOTOR [16] and CLMBR [17]), benchmark releases (EHRSHOT [22] and INSPECT [5]), and evaluation of model performance across different hospitals [4].

**Event Stream GPT (ESGPT)** ESGPT [12] contains a data pre-processing and extraction system that can produce MEDS-compliant datasets from various raw source files. Furthermore, their default fully generative, autoregressive neural network foundation model architectures can likewise be trained on MEDS datasets.

**General Healthcare Predictive Framework (GenHPF)** GenHPF [6] provides a set of transformation functions that converts MEDS-compliant datasets into GenHPF-formatted datasets that can be directly processed to GenHPF framework. In addition, by utilizing ESGPT as an extractor of MEDS, they provide a data pre-processing pipeline that can generate example MEDS-compliant datasets for three different public EHR datasets: MIMIC-III [9], MIMIC-IV [8], and eICU [14].

**Yet Another ICU Benchmark (YAIB)** YAIB [19] is an end-to-end ICU benchmarking framework designed for creating experiments over multiple cohorts from a range of ICU datasets. MEDS-compliant datasets can be used as an alternative data input format for benchmarking clinical prediction tasks such as Sepsis, Mortality, and Acute Kidney Injury. A range of (extendable) ML and DL models can be used for experiments.

## 4 Discussion

### 4.1 Related Works

Several initiatives have been proposed to standardize EHR data across hospitals by defining a "common data model" (CDM). The Observational Medical Outcomes Partnership Common Data Model (OMOP-CDM) was developed by the Observational Health Data Sciences and Informatics (OHDSI) Consortium for observational health studies, and is currently used by over 90 health systems [10]. It defines a set of 394 fields over 39 relational tables which capture a condensed, patient-level view of a hospital's underlying EHR. The PCORNet CDM defines a set of 15 tables that have been adopted by roughly 80 clinical sites to enable nation-wide comparative effectiveness research by supporting SQL/SAS queries from a coordinating center [10]. Informatics for Integrating Biology and the Bedside (i2b2) is another clinical data warehousing standard currently used by over 200 sites. It offers a more flexible standard than PCORNet and OMOP-CDM and comes with a robust analytics platform.

There are a number of Python libraries that exist for training ML models on EHR data. PyHealth is an end-to-end deep learning framework for EHRs, and natively supports MIMIC, eICU, and OMOP-CDM data sources in addition to providing a number of ML models out-of-the-box [23]. As it leverages pandas dataframes to represent data internally, PyHealth represents objects at the table- rather than patient-level, and does not readily scale to millions of patients. TemporAI [15] and Clairvoyance [7] are geared towards medical time-series data, with particular attention to survival analysis and prediction respectively. However, they lack have native support for the coded ontologies that are endemic to structured EHR data. YAIB is a benchmark for ICU tasks that can ingest MIMIC, eICU, HiRID, and AUMCdb under a shared interface, but is limited to the ICU setting [19]. FIDDLE is a preprocessing library that generates feature vectors for downstream ML models given tabular EHR data [18].

### 4.2 Limitations & Future Work

There are several limitations to MEDS. First, data is stored in a patient-centric manner, i.e. the information is joined on each unique patient. This makes MEDS ill-suited for calculating summary statistics aggregated across a cohort of patients (e.g. average patient age or total count of a specific code), and we instead recommend that such analyses be computed externally to MEDS (e.g. via SQL on the original data source or by expanding the MEDS format into a code- or event-centric representation via other data processing tools) or via online algorithms while iterating over patients. Second, MEDS is simply a specification – it does not actually provide any code for implementing the standard. In the same way that JSON is a data standard that has separate implementations across different programming languages, we anticipate that the community will develop different frameworks for implementing MEDS that tailor it for specific use cases. Several such efforts are already underway, such as FEMR [22], ESGPT [12], and GenHPF [6], and YAIB [19] are unified by their shared reliance on MEDS as their underlying data representation.

Going forward, we envision MEDS serving as the bedrock for an EHR-native deep learning ecosystem which brings the same ease-of-use and composability to healthcare data that frameworks like HuggingFace have brought for NLP. This will eventually include software tools and best practices for model development, training, and evaluation, all of which will emerge organically through community use. We hope these efforts bring together a community of researchers excited by the prospect of bringing the latest advancements in deep learning to healthcare, and united by a set of shared best practices and standards which are currently absent from the field.

## References

[1] Omop common data model. `https://www.ohdsi.org/data-standardization/`. Accessed: 2023.

[2] Duane Bender and Kamran Sartipi. Hl7 fhir: An agile and restful approach to healthcare information exchange. In *Proceedings of the 26th IEEE international symposium on computer-based medical systems*, pages 326–331. IEEE, 2013.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] Lin Lawrence Guo, Jason Fries, Ethan Steinberg, Scott Lanyon Fleming, Keith Morse, Catherine Aftandilian, Jose Posada, Nigam Shah, and Lillian Sung. A multi-center study on the adaptability of a shared foundation model for electronic health records. *arXiv preprint arXiv:2311.11483*, 2023.

[5] Shih-Cheng Huang, Zepeng Huo, Ethan Steinberg, Chia-Chun Chiang, Matthew P. Lungren, Curtis P. Langlotz, Serena Yeung, Nigam H. Shah, and Jason A. Fries. Inspect: A multimodal dataset for pulmonary embolism diagnosis and prognosis, 2023.

[6] Kyunghoon Hur, Jungwoo Oh, Junu Kim, Jiyoun Kim, Min Jae Lee, Eunbyeol Cho, Seong-Eun Moon, Young-Hak Kim, Louis Atallah, and Edward Choi. Genhpf: General healthcare predictive framework for multi-task multi-source learning. *IEEE Journal of Biomedical and Health Informatics*, 2023.

[7] Daniel Jarrett, Jinsung Yoon, Ioana Bica, Zhaozhi Qian, Ari Ercole, and Mihaela van der Schaar. Clairvoyance: A pipeline toolkit for medical time series. In *International Conference on Learning Representations*, 2021.

[8] Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1, 2023.

[9] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[10] Jeffrey G Klann, Lori C Phillips, Christopher Herrick, Matthew AH Joss, Kavishwar B Wagholikar, and Shawn N Murphy. Web services for data warehouses: Omop and pcornet on i2b2. *Journal of the American Medical Informatics Association*, 25(10):1331–1338, 2018.

[11] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery.

[12] Matthew McDermott, Bret Nestor, Peniel Argaw, and Isaac Kohane. Event stream gpt: A data pre-processing and modeling library for generative, pre-trained transformers over continuous-time sequences of complex events. *arXiv preprint arXiv:2306.11547*, 2023.

[13] Matthew B. A. McDermott, Shirly Wang, Nikki Marinsek, Rajesh Ranganath, Luca Foschini, and Marzyeh Ghassemi. Reproducibility in machine learning for health research: Still a ways to go. *Science Translational Medicine*, 13(586):eabb1655, 2021.

[14] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5(1):1–13, 2018.

[15] Evgeny S Saveliev and Mihaela van der Schaar. Temporai: Facilitating machine learning innovation in time domain tasks for medicine. *arXiv preprint arXiv:2301.12260*, 2023.

[16] Ethan Steinberg, Jason Fries, Yizhe Xu, and Nigam Shah. Motor: A time-to-event foundation model for structured medical records, 2023.

[17] Ethan Steinberg, Ken Jung, Jason A. Fries, Conor K. Corbin, Stephen R. Pfohl, and Nigam H. Shah. Language models are an effective representation learning technique for electronic health record data. *Journal of Biomedical Informatics*, 113:103637, 2021.

[18] Shengpu Tang, Parmida Davarmanesh, Yanmeng Song, Danai Koutra, Michael W Sjoding, and Jenna Wiens. Democratizing ehr analyses with fiddle: a flexible data-driven preprocessing pipeline for structured clinical data. *Journal of the American Medical Informatics Association*, 27(12):1921–1934, 2020.

[19] Robin van de Water, Hendrik Schmidt, Paul Elbers, Patrick Thoral, Bert Arnrich, and Patrick Rockenschaub. Yet another icu benchmark: A flexible multi-center framework for clinical ml. June 2023. arXiv:2306.05109 [cs].

[20] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[21] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[22] Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason A. Fries, and Nigam H. Shah. Ehrshot: An ehr benchmark for few-shot evaluation of foundation models, 2023.

[23] Chaoqi Yang, Zhenbang Wu, Patrick Jiang, Zhen Lin, Junyi Gao, Benjamin Danek, and Jimeng Sun. PyHealth: A deep learning toolkit for healthcare predictive modeling. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) 2023*, 2023.