Towards A Platform and Benchmark Suite for Model Training on Dynamic Datasets

Maximilian Böther
ETH Zurich
Switzerland
mboether@inf.ethz.ch

Foteini Strati
ETH Zurich
Switzerland
fstrati@inf.ethz.ch

Viktor Gsteiger
ETH Zurich
Switzerland
vgsteiger@student.ethz.ch

Ana Klimovic
ETH Zurich
Switzerland
aklimovic@ethz.ch

ABSTRACT

Machine learning (ML) is often applied in use cases where training data evolves and/or grows over time. Training must incorporate data changes for high model quality, however this is often challenging and expensive due to large datasets and models. In contrast, ML researchers often train and evaluate ML models on static datasets or with artificial assumptions about data dynamics. This gap between research and practice is largely due to *(i)* the absence of an open-source platform that manages dynamic datasets at scale and supports pluggable policies for when and what data to train on, and *(ii)* the lack of representative open-source benchmarks for ML training on dynamic datasets. To address this gap, we propose to design a platform that enables ML researchers and practitioners to explore training and data selection policies, while alleviating the burdens of managing large dynamic datasets and orchestrating recurring training jobs. We also propose to build an accompanying benchmark suite that integrates public dynamic datasets and ML models from a variety of representative use cases.

ACM Reference Format:
Maximilian Böther, Foteini Strati, Viktor Gsteiger, and Ana Klimovic. 2023. Towards A Platform and Benchmark Suite for Model Training on Dynamic Datasets. In *3rd Workshop on Machine Learning and Systems (EuroMLSys '23), May 8, 2023, Rome, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3578356.3592585

1 INTRODUCTION

Machine Learning (ML) datasets encountered in real-life scenarios are inherently dynamic, i.e., samples get added or removed over time. Data is constantly collected via sensors, IoT devices, self-driving cars, and satellites [8, 40, 69]. At Meta, the training dataset used for recommendation models spans *exabytes* of storage, and its size has *doubled* in the past two years [81]. At the same time, social trends and emerging events cause natural data distribution shifts [38, 71]. Data may also need to be deleted after a period of time, for example, due to privacy regulations, such as the GDPR [21].

ML models need to account for the dynamic nature of data. According to Meta's studies, training recommendation models daily results in significantly higher quality than training weekly [26, 81]. Similarly, transformer models struggle to keep up with the distribution shifts in real datasets, requiring frequent updates [38].

Training ML models on dynamic datasets raises two key questions: *when* to update an ML model, and *what* data to train it on? Naively retraining a model from scratch on the entire dataset when new data becomes available is prohibitively expensive and slow. The training cost is proportional to the data volume, which can be in the order of petabytes or exabytes [22, 81]. Fine-tuning the model on each new example as it arrives is also impractical in production, due to strict, time-consuming deployment checks [29].

The ML community has been exploring these questions by developing techniques to detect data distribution shifts [40, 58] and find an appropriate subset of data to train on [33, 46, 55, 56]. However, most of these works focus on small-scale, static datasets such as CIFAR [37], MNIST [39], and ImageNet [17], while datasets used in production are significantly larger and dynamic. Previous works often artificially split data into tasks, and gradually reveal them to the model to simulate dynamicity [56]. Unfortunately, these assumptions are not always realistic [78], leaving a gap between research and practice.

We identify two key reasons for the gap between practical ML use cases and the scenarios typically studied by researchers. First, there is no end-to-end ML platform built from the ground up for dynamic datasets. Such a platform needs to deal with huge amounts of data and metadata, monitor the data ingestion process, and implement and apply the appropriate data selection algorithm, which often has a high computational cost. The platform must also be modular and flexible enough to encompass various use cases with different policies for when and what to train on. Although the need for such a platform has been acknowledged by ML

EuroMLSys '23, May 8, 2023, Rome, Italy

ACM ISBN 979-8-4007-0084-2/23/05.
https://doi.org/10.1145/3578356.3592585

practitioners [52, 61, 70], and there exist various individual components for different parts of this process [6, 49, 74], there is currently no open-source platform that meets all these requirements. Second, it is difficult to find representative open-source ML benchmarks with dynamic datasets, as most companies keep their data private for regulatory and tradesecret reasons. This causes academic research to evolve around small-scale, static datasets. In this paper, we outline our vision on how to address these two problems. We make the following contributions.

(1) We propose to bring often separated areas of ML research together by formulating the paradigm of ML training on dynamic datasets. We discuss how existing research maps onto the two dimensions, *when* and on *what* data to train.
(2) We present our vision for an open-source end-to-end platform for ML on dynamic datasets, Modyn, which aims to enable ML researchers to easily experiment with data selection and training triggering policies, while abstracting the complexities of large-scale data management and system orchestration.
(3) To facilitate ML research on practical use cases, we propose a benchmark suite comprising open-source datasets that are inherently dynamic and span various application domains (e.g., recommendation systems, autonomous driving, NLP). We are in the process of combining these datasets with publicly available models, and integrating them in an initial prototype of Modyn to allow for end-to-end, easy-to-use, representative benchmarks.

This paper presents early-stage work and lays out our initial ideas on how to address model training on dynamic datasets. We welcome feedback and contributions from the community. We aim to stimulate collaboration between industry and academia to explore diverse challenges that arise in practical ML use cases, where data is inherently dynamic.

2 BACKGROUND AND MOTIVATION

We define dynamic datasets (Section 2.1) and outline the challenges they present from a ML theory perspective (Section 2.2) and system perspective (Section 2.3). The gap in system support for managing dynamic data in today's ML ecosystem motivates us to propose a new platform and benchmark suite for ML on dynamic datasets.

2.1 Dynamic Datasets

We define *dynamic datasets* as datasets that *change over time*. Changes may consist of adding new data points (e.g., a sensor data source continually collects new data), removing old data points (e.g., due to privacy regulations), or editing data points (e.g., by applying data cleaning techniques). In real-world applications of ML, training data is often dynamic. In contrast, ML researchers often evaluate ML models and training algorithms using *static datasets*, which remain fixed over time, such as ImageNet [17], MNIST [39], or CIFAR [37].

Implications for ML. Machine learning models should account for the dynamic nature of data for three key reasons. First, new data reflects the latest trends and reveals potential *distribution shifts*, which are critical to capture in many application domains, such as recommender systems [25, 26]. For example, in GrubHub's food delivery service, incorporating data from a day before model training increases the purchase rate by 20 % compared to training on data from only the previous week [20]. Second, even if data distributions remain fairly stable over time, simply training on *more* data as it becomes available can improve model quality, by allowing the model to generalize across more data points. In many real-world settings, data is continuously collected. For example, Tesla continuously captures new image data from streets all over the world and uses this data to train their autonomous driving models to generalize better [69]. Finally, model training also needs to comply with privacy regulations (e.g., GDPR [21], CCPA [63]), which may require some data to be deleted after a period of time. Deleted data must be "unlearned" by the model [10].

Hence, ML practitioners frequently retrain or finetune models to account for dataset updates [6, 28–30, 36, 52, 70]. As datasets grow to petabyte scale [81], frequently revisiting all data for training is cost and time prohibitive.

2.2 ML Theory Perspective

Two key questions arise for efficient ML training ML on dynamic datasets: *when* to (re)train a model and on *what* data? ML researchers have proposed techniques for when to train [24, 40, 44, 58, 67, 68] and what data to train on [1, 2, 31, 33, 34, 45, 46, 55, 56]. However, these research questions have thus far primarily been explored separately and in the context of static datasets. For example, the data selection community has focused on finding representative subsets of data for static datasets, which can be used to train a ML model with fewer data samples and hence lower cost at similar accuracy. Meanwhile, statisticians have explored how to quantify and detect shifts in data distributions, which can be useful for determining when a model should train on new data.

Continual learning (CL) [1, 2, 4, 19, 34, 42, 56]—also called incremental learning [13, 54, 77]—is a related paradigm of ML training on continuous streams of data. CL focuses on learning new classes, usually grouped into tasks. CL has different variants and assumptions, such as the disjoint task formulation, stating that the model only sees data for task 1, then only data for task 2, etc. A common benchmark dataset is Split-MNIST [1, 56], which defines a task as two numbers in MNIST. However, this and other datasets commonly used for CL research are synthetic, small, and lack a real notion of

time. This makes them unrepresentative of dynamic datasets that arise in practical ML use cases in the wild, where it is often necessary to learn *new representations* of existing classes rather than new classes [78]. Furthermore, CL research has not addressed the question of when to trigger training.

2.3 (Lack of) System Support

Requirements. A platform for ML on dynamic datasets must support frequent model (re)training, by implementing algorithms for when and what data to train on. The platform must also store large volumes of data at low cost while providing fast access to the most relevant data for training. As algorithms deciding what data to train on often rely on information from previous training rounds, such as the loss [45], or pre-computed forward passes to upper bound the gradient norm [33], the platform must also manage large volumes of metadata. For example, training a recommendation model on the 1TB Criteo dataset [16], which contains 24 days worth of data with 180 million samples per day, requires logging metadata for 4 320 million sample IDs.

Current Solutions. ML operations (MLOps) refers to system infrastructure for managing ML workflows and model life cycles [72]. Several closed-source platforms, like Weights & Biases [9], support data selection and dataset versioning [74]. Other relevant commercial systems include Amazon SageMaker [3] and NeptuneAI [49]. Open-source platforms such as Tensorflow TFX [47] and MLflow [14] support data validation, model versioning, and experiment tracking, but were not built with dynamic datasets in mind. Setting up a workflow that frequently trains a model requires significant manual plumbing [6]. To ingest dynamic data from stream processing engines and batch ETL jobs, practitioners can use feature stores, such as Feast [23], which build, store, and update features from raw data for training or inference. Renate [80] and Avalanche [41] offer experimental infrastructure for continual learning policies. Systems such as Ekya [8], which optimizes continuous retraining for vision models on edge devices, and Ekko [62], which optimizes the propagation of model updates for recommendation systems, cater to specific use cases. In contrast, we strive for a more general platform to explore data selection and training policies across a variety of application domains. While current solutions offer useful building blocks, there is no off-the-shelf end-to-end platform that meets these requirements.

Call To Action. To enable the exploration of research challenges that arise for ML on dynamic datasets, we need an end-to-end platform that seamlessly manages large volumes of data and metadata, while providing pluggable interfaces for the algorithmic exploration of when to trigger model training and on which data (Section 3). Researchers also need access to representative open-source datasets and model training benchmarks (Section 5).

3 DESIGNING A PLATFORM FOR ML ON DYNAMIC DATASETS

We first formulate the paradigm of ML training on dynamic datasets (Section 3.1). We then present an initial architecture for a system supporting this paradigm (Section 3.2).

3.1 Modeling the Training Process

When training on dynamic datasets, there are two crucial decisions, namely *(i)* determining the best time *when* to initiate a new training process, and *(ii)* selecting *what* is the most appropriate data to train on. We refer to the initiation of a new training process as a *trigger*, and the decision process involved in determining the optimal time for starting a new training process as the *triggering policy*. Notably, the decision on what data to use for training is independent of when to trigger, and it is governed by a *data selection policy*.

Triggering. Consider a discrete time clock that governs the arrival of a list of $n_t \geq 0$ new samples denoted as $S_t = (s_1, ..., s_{n_t})$, where t is the current timestep. Each sample $s_i \in S_t$ comprises a unique identifier, a label, and a data payload. We provide S_t to the triggering policy upon arrival. The policy's objective is to determine which $s_i \in S_t$ triggers a new training process, if any, i.e., it outputs an ordered list $\mathcal{T}_t = (i \in [1 \dots n_t] \mid s_i \in S_t \text{ causes trigger})$. The triggering policy is stateful and can theoretically utilize information over the entire history of samples, i.e., for all $t' \leq t$, it can leverage $S_{t'}$ or properties of it, to come to a triggering decision. For instance, the policy could choose to trigger retraining every 10th data point by tracking the number of data points observed since the last trigger.

Data Selection. On each trigger, the data selection policy selects which data points to train on. The policy selects from all previously seen data points, i.e., they can come from any $S_{t'}$ with $t' < t$, and all samples in S_t until the trigger occurs. Each trigger has a unique, strictly monotonously increasing id $k \in \mathbb{N}$. Let $k \in \mathcal{T}_t$ with $s_k \in S_t$ causing the overall x-th trigger. The data selection policy defines and provides the x-th *trigger training set*

$$\mathcal{D}_x \subseteq \left(\{s_i \in S_t \mid i \leq k\} \cup \bigcup_{t'<t} \text{set}\,(S_{t'}) \right) \times \mathbb{R}, \qquad (1)$$

i.e., a set of samples and associated weights to train on for trigger x. The sample weights are used to prioritize samples by multiplying their gradients with the weights during backpropagation. The trigger training set is a *subset* of all data points seen so far, so it may, but does not have to, contain samples from previous triggers. The details on how $\mathcal{D}_x$ is calculated are algorithm-dependent.

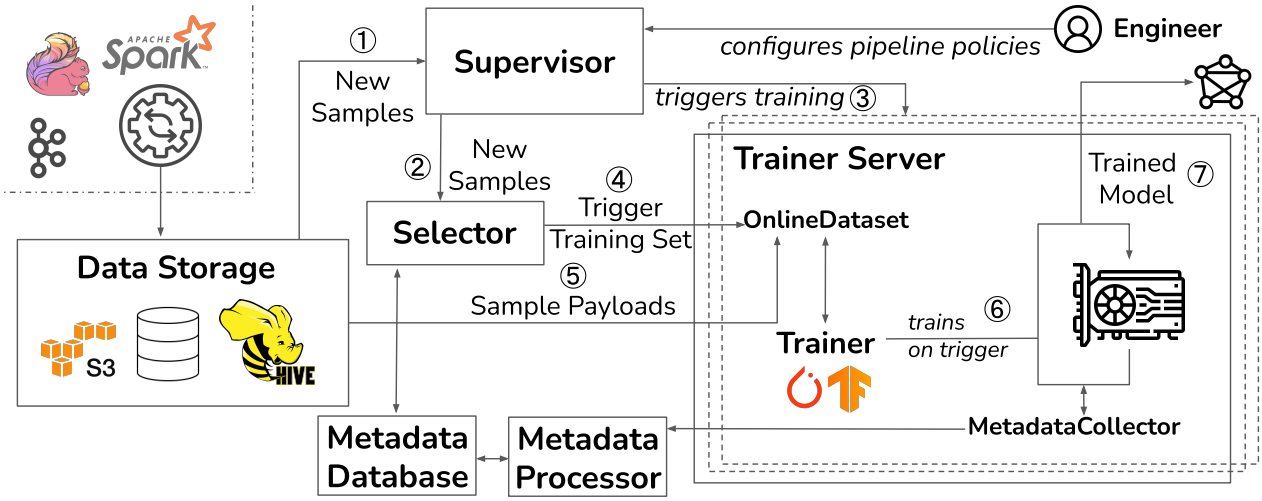

Figure 1: Proposed Modyn system architecture.

3.2 Platform Architecture

We present our vision for the Modyn platform and an early-stage proof of concept[1]. In our discussion here, we are not concerned with proposing an optimized system design. Rather we discuss a high-level system architecture to give a concrete idea of what a platform that addresses the requirements for model training on dynamic datasets might look like.

Figure 1 shows the proposed system architecture. The core of Modyn consists of the *supervisor*, which implements the triggering policy, and the *selector*, which implements the data selection policy, based on metadata collected and managed by the platform. The trainer server executes training jobs on data fetched from Modyn's data storage component. For modularity, all components are decoupled and communicate via gRPC and FTP. Section 3.4 describes each component's role in more detail.

We assume Modyn ingests data from stream processing engines (e.g., Flink [12]) or batch processing frameworks (e.g., Spark [79]). We expect a *labled* input data stream. Such labels can be either obtained automatically (e.g., track which advertisements a user clicked on) or from human-in-the-loop annotation systems [76] that provide end-to-end solutions for data labeling. The output of Modyn is a stream of trained ML models that can then either be further processed or deployed for serving, using tools like BentoML [7], TorchServe [57], or Triton Inference Server [51].

Modyn Pipelines. The core unit of execution in Modyn is a *pipeline*. A pipeline comprises a comprehensive description of a training process on a dynamic dataset, i.e., it defines the trigger policy, the data selection policy, the model architecture, and further training parameters, such as learning rate, optimization criterion, etc. In the prototype, it is configured via a yaml file (c.f. Section 3.3). To run a pipeline, the user starts the supervisor and supplies the configuration file.

Overview of Data Flow. When running a pipeline, data samples stream in from outside Modyn into the data storage component, which assigns a key to each sample that is used to uniquely identify it. Data storage informs the supervisor about new samples by their key ①. The supervisor checks for triggers and forwards potential triggers and the sample keys to the selector ②. Upon trigger, the supervisor contacts the trainer server to start a training process ③. The trainer requests the trigger training set from the selector ④, and the sample data from the storage ⑤. The trainer then runs a training according to the configuration ⑥. The trained model, the output of Modyn, can then be used in further steps in the overall ML workflow, such as deployment ⑦.

Current Prototype. Our proof of concept supports ML pipelines with custom optimizers, learning rate schedulers, mixed-precision training, custom CUDA extensions, and other features needed to run an initial example use case of recommender system training, described in Section 4 We can execute pipelines in either *experiment mode* or *production mode*. In production mode, the data storage informs the supervisor when new data points come in. In experiment mode, the data storage simulates new data points streaming in by announcing already existing data points as "new" to the supervisor. The experiment mode can be used to play various traces and compare how policies do given the same data environment and initial model config. The insights gained from these experiments can then be used to find a configuration for production mode.

[1]Our code is available at: https://github.com/eth-easl/modyn.

```yaml
model:
  id: ResNet18
  config: ...
training:
  use_previous_model: True
  optimizers: ...
  optimization_criterion:
    name: "CrossEntropyLoss"
  selection_strategy:
    name: NewDataStrategy
    config:
      reset_after_trigger: True
data:
  dataset_id: mnist
  transformations: ["transforms.Normalize(...)"]
  bytes_parser_function: |
    def bytes_parser_function(data: bytes) -> Image:
      return Image.open(io.BytesIO(data))
trigger:
  id: DataAmountTrigger
  trigger_config:
    data_points_for_trigger: 100
```

Listing 1: Excerpt from a example Modyn pipeline

3.3 Pipeline User API

A pipeline is the core unit of execution in Modyn. Listing 1 shows an example pipeline. At minimum, a pipeline consists of *(1)* the model specification, *(2)* the training dataset, and a corresponding byte parsing function that defines how to convert raw sample bytes to model input, *(3)* the trigger policy, *(4)* the data selection policy, *(5)* training hyperparameters such as optimization criterion, optimizer, learning rate, batch size, and *(6)* training configuration such as data processing workers, whether to use automatic mixed precision, etc. The user can define whether a training on trigger should start with the previously trained model, or start with a randomly initialized model each time. The very first training can run on a randomly initialized or externally provided model. When a model is supported by Modyn, a ML engineer can run a pipeline by simply providing a yaml file like the one in Listing 1. Users can easily add new models, triggering policies, and data selection strategies as pluggable Python modules and use the yaml configuration file to select particular strategies for a pipeline.

3.4 Component Overview

We discuss each Modyn component from Figure 1, including its role in enabling ML researchers to explore triggering and data selection policies while alleviating users from the burden of data management and job orchestration.

Supervisor. The supervisor is the brain of a pipeline as it coordinates the control and data flow. It registers a new pipeline at all components. The supervisor receives the keys of new samples from storage, and forwards them to the selector. On the x-th trigger, it notifies the trainer server to train on the trigger training set $\mathcal{D}_x$. The supervisor implements additional coordination mechanisms, such as supplying the model from the previous trigger to the trainer server.

In addition to coordinating pipeline execution, the supervisor implements triggering policies. We identify three types of triggers: *(i)* amount-based, *(ii)* time-based, and *(iii)* drift-based triggers. Amount-based triggers fire every n data points, while time-based triggers fire after a time interval has passed. Drift-based triggers are based on detection of distribution shifts, either in the input data distribution or the model output distribution (i.e., the performance of the deployed model). They are more adaptive to the data stream than amount- and time-based triggers. We currently support time and amount-based triggers and plan to add support for drift-based triggers. For input-drift triggers, we can leverage research from the ML community on distribution shift detection, with methods such as MatchMaker [44], Odin [67], DriftSurf [68], and many more [24, 40, 58]. Output-drift triggers require connecting Modyn to the inference pipeline, which our prototype does not currently support.

Selector. The selector implements data selection policies that, on trigger x, output the trigger training set $\mathcal{D}_x$. We differentiate between online and offline selection policies. Offline policies are policies that calculate $\mathcal{D}_x$ on trigger by collecting all samples and running calculations on potentially all previous data points. Common examples of such policies include *coreset* algorithms. While the term coreset can be generally used to describe any data reduction technique, the approach has primarily been applied to static datasets. Examples of offline coreset algorithms include DLIS [33], CRAIG [46], and AdaCore [55]. Examples of other offline selection techniques include RHO-LOSS [45] and Shapley-value based techniques [31]. Examples of online policies include continual learning samplers, such as GDumb [56], MIR [1], CLiB [34], and GSS [2]. The pluggable selector interface aims to enable ML researchers to implement and compare such policies, as well as develop their own policies to optimize model accuracy, training time, energy, and cost.

In our proof of concept, we implement simple baseline strategies, such as outputting all seen samples, subsampling a smaller data set uniformly at random, or subsampling with priority on newer data points. Data selection strategies can be used with or without *reset* of the internal strategy state after a trigger, e.g., the strategy that outputs all seen datapoints can be used both for retraining from scratch (no reset) or finetuning the previous model (reset).

Trainer Server. Compatibility with existing ML infrastructure is one of our design principles. The trainer server implements a generic training interface that runs a model training job based on a pipeline definition. Our prototype currently implements a PyTorch-based trainer, however, the design is agnostic to the ML framework and can be extended to

Tensorflow or higher level abstractions, such as SageMaker. The trainer server executes a training loop, which consists of steps, such as model initialization, setup of optimizer(s), and handling registered callbacks (e.g., for metadata collection). The trainer fetches data from the storage layer and interacts with the selector via an `OnlineDataset` abstraction.

Data Storage. Modyn pipelines ingests files from data sources such as stream processing or ETL jobs. The data storage component abstracts different file formats and file systems (e.g., local file system or remote storage services like S3). Each file can contain one or more samples, which can be of any form, e.g., entire files, database rows, etc. The goal of Modyn's storage abstraction is to decouple the trainer from the underlying data representation and to minimize the amount of data transferred. Our proof of concept currently assumes data is stored in a flat, single-tier storage system. Extending the storage hierarchy to multiple tiers and designing data importance-aware caching policies is promising direction to optimize data storage costs and performance.

Metadata Processor. The metadata processor is responsible for transforming collected metadata (e.g., compressing gradients) and persisting it to Modyn's metadata database. While we currently collect and store metadata per training sample, the optimal granularity for metadata collection and storage to balance storage costs, query latencies, and the effectiveness of data selection and triggering policies remains an open question.

4 INITIAL EXPERIENCE

Example Use Case. We train a DLRM recommendation model [48] on the Criteo 1TB click stream dataset [16], which provides user data over 24 days, with roughly 180 million samples per day. As the dataset is split into days, we can replay it as a dynamic dataset over time. Given anonymized categorical and numerical features of users, the task is to predict whether a user will click on an advertisement or not.

Experimental Setup. We integrate NVIDIA's implementation [50] of a DLRM [48] in the Modyn prototype. Following NVIDIA's small setup [50], we preprocess the Criteo dataset with a frequency threshold of 15. We configure the pipeline with the training hyperparameters (optimizers, learning rate scheduler) in the NVIDIA repository and use mixed-precision training. We train on a A100 40 GB GPU with a time-based daily trigger, i.e., we trigger training for each day in the dataset, starting with the model from the previous trigger. This means that we finetune the model from the previous day on the subsequent day.

Initial Results. In Figure 2, we show the area under curve (AUC) for the models trained per daily trigger from day 1 to day 9 and evaluating on day 10. Training on fresher data is beneficial, as the AUC increases by 3 percentage points

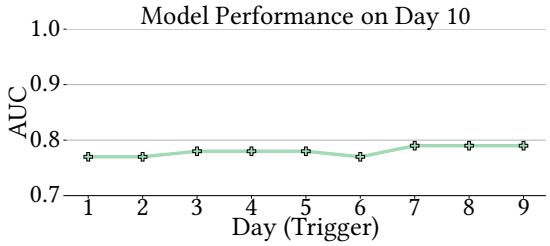

Figure 2: DLRM performance on day 10 of Criteo, when finetuned on data from day 0 - day 9.

from 0.76 for day 1 to 0.79 for day 9. Considering that recent work reports that 0.2 percent points increase in CTR AUC leads to a 1% annual revenue increase [75], these results seem promising. This experiment motivates further research on the performance and cost-performance tradeoffs of triggering and selection policies, which we discuss in Section 6.

5 TOWARDS A BENCHMARK SUITE

Previous work on data selection and drift detection has focused on static, small-scale datasets, which do not reflect many real scenarios. Accompanying Modyn, we are in the process of building a benchmark suite for representative ML use cases with dynamic datasets. We focus on publicly-available datasets which are inherently dynamic and tasks where ML models need to be adjusted in order to maintain good quality. Our goal is to fully integrate these datasets and models in Modyn to facilitate future research. Next, we describe some use cases we plan to include.

Recommendation Systems. Clickthrough rate (CTR) prediction is a common use case for recommendation models. We integrate the Criteo 1 TB 24-day dataset [16] and the 10-day Avazu [73] dataset with NVIDIA's DLRM implementation [48, 50].

Natural Language Processing. We target content classification with NLP models, such as BERT [18], on a collection of Reddit data gathered with the PushShift API [5]. The task is to classify posts into subreddits. To achieve high accuracy, the model must adjust to topic discussions over time [38, 78].

Autonomous Driving. We target motion prediction in self-driving cars, integrating datasets such as Waymo [66], nuScenes [11], and Cityscapes [15], along with open-source object detection models, such as YOLO [59]. As autonomous driving data is continuously collected to capture as many scenarios as possible, models need to be frequently updated [22, 40, 69].

Weather Prediction. We focus on atmospheric variable prediction using the ERA5 [27] dataset, with the FourCastNet [53] model. The ERA5 dataset contains multiple climate features (e.g., moisture, wind, radiation, precipitation) gathered over many years. The natural temporal distribution shift of weather over time requires model adjustments [40, 78].

The ML community has developed several benchmarks for data distribution shifts, which we can also incorporate. WILDS [35] spans 10 datasets with applications such as healthcare and toxicity classification. Wild-Time [78] consists of 5 datasets with temporal distribution shifts, encompassing use cases such as healthcare and news classification. Shifts [43] provides industry-sourced datasets, including weather prediction, machine translation, and self-driving cars. Vela et al. [71] analyze model performance degradation over time on 32 datasets from 4 industries. We welcome contributions to the benchmark suite.

6 OPEN RESEARCH QUESTIONS

We discuss open research questions that arise for ML on dynamic datasets from both a systems and ML perspective.

Designing Data Selection and Triggering Policies. While ML literature has explored data selection for both static and dynamic datasets [1, 33, 34, 45, 56], the adaption of existing static methods, such as coresets [33, 46, 55], to the dynamic setting remains an open problem. Similarly, while prior work explores how data drift affects model accuracy [20, 26, 38, 81], designing training triggering policies–particularly in concert with data selection policies–to address changes in training data distributions is an open research area. In context of system/algorithm co-design, feedback from inference can help detect when to trigger the next training (e.g., when inference performance degrades).

Metrics for Policy Comparison. Triggering and data selection policies impact the frequency of training and how much data is selected for a training pass. Reasoning about the joint impact on training time, training cost, energy consumption, and model accuracy is non-trivial. What are the right metrics to quantify and reason about the accuracy-energy-cost tradeoff of data selection and triggering policies across a variety of ML training use cases?

Metadata Management. Which metadata should be persisted and how to store it efficiently? It is not clear at which granularity metadata needs to be maintained to effectively support data selection and triggering policies. Managing per-sample metadata for large datasets with billions of samples while minimizing storage costs and enabling low-latency queries is challenging. How can we efficiently collect and store metadata such that metadata collection and querying is not a bottleneck during training?

Model Management. Model management is an active area of research [60, 64, 65]. Multiple model versions naturally arise when training jobs are continually triggered and it may be desirable to switch between different models (e.g., exploration of different models or rollback in case of performance degradation). How can we optimize model storage and retrieval? For example, when the weights of a model change only slightly between triggers, we could store only the changes to the weights to reduce storage requirements.

Monitoring and Debugging. ML models can fail in complex ways, making continuous model monitoring and testing critical for production use cases. Debugging model performance is challenging [3, 32], and is only made more difficult when models are frequently updated in response to changes in training data. What kind of system support is needed for model performance debugging, particularly in dynamic data environments?

7 CONCLUSION

We identify a gap between ML research and practice: while researchers often train and evaluate ML models with static datasets, ML models in the wild often train on large datasets that are dynamic in nature. Two key challenges arise for ML on dynamic datasets: *when* to train and on *what* data to train). However today's ML ecosystem lacks infrastructure to jointly explore these questions with representative use cases. We propose our vision for an open-source platform and benchmark suite to explore training triggers and data selection policies in practical use cases, while alleviating users from the burden of managing large datasets and orchestrating recurring training jobs. Model training on dynamic data opens many research directions, such as designing efficient metadata and data management systems and designing data selection and triggering policies to meet model accuracy requirements while minimizing training costs. We welcome contributions from the community.

ACKNOWLEDGMENTS

We thank Ambarish Prakash, Roxana Stiuca, and Kevin Shao for their contributions towards the prototype codebase. We thank Theo Rekatsinas, Newsha Ardalani, Benoit Steiner, Ville Kallioniemi, and Jiří Šimša for insightful discussions. Maximilian Böther and Foteini Strati are supported by the Swiss National Science Foundation (Project Number 200021_204620). We are grateful for Google Cloud credits.

REFERENCES

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. 2019. Online Continual Learning with Maximal Interfered Retrieval. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2019/hash/15825aee15eb335cc13f9b559f166ee8-Abstract.html

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. 2019. Gradient based sample selection for online continual learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2019/hash/e562cd9c0768d5464b64cf61da7fc6bb-Abstract.html

[3] Amazon. 2023. Amazon SageMaker. https://docs.aws.amazon.com/sagemaker/index.html.

[4] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. 2021. Rainbow Memory: Continual Learning with a Memory of Diverse Samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr46437.2021.00812

[5] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The Pushshift Reddit Dataset. In *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*. https://doi.org/10.48550/ARXIV.2001.08435

[6] Denis Baylor, Kevin Haas, Konstantinos Katsiapis, Sammy Leong, Rose Liu, Clemens Mewald, Hui Miao, Neoklis Polyzotis, Mitchell Trott, and Martin Zinkevich. 2019. Continuous Training for Production ML in the TensorFlow Extended (TFX) Platform. In *Proceedings of the USENIX Conference on Operational Machine Learning (OpML)*. https://www.usenix.org/conference/opml19/presentation/baylor

[7] BentoML. 2023. BentoML: Github Organization. https://github.com/bentoml/. Accessed: 2023-03-09.

[8] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. 2022. Ekya: Continuous Learning of Video Analytics Models on Edge Compute Servers. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. https://www.usenix.org/conference/nsdi22/presentation/bhardwaj

[9] Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. https://www.wandb.com/

[10] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine Unlearning. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/sp40001.2021.00019

[11] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr42600.2020.01164

[12] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache Flink™: Stream and Batch Processing in a Single Engine. *Bulletin of the Technical Committee on Data Engineering* 38, 4 (2015).

[13] Gert Cauwenberghs and Tomaso A. Poggio. 2000. Incremental and Decremental Support Vector Machine Learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2000/hash/155fa09596c7e18e50b58eb7e0c6ccb4-Abstract.html

[14] Andrew Chen, Andy Chow, Aaron Davidson, Arjun DCunha, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Clemens Mewald, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Avesh Singh, Fen Xie, Matei Zaharia, Richard Zang, Juntai Zheng, and Corey Zumar. 2020. Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle. In *Proceedings of the International Workshop on Data Management for End-to-End Machine Learning (DEEM)*. https://doi.org/10.1145/3399579.3399867

[15] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. https://doi.org/10.1109/cvpr.2016.350

[16] Criteo. 2013. Download Terabyte Click Logs. https://labs.criteo.com/2013/12/download-terabyte-click-logs/.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. https://doi.org/10.18653/v1/n19-1423

[19] Tom Diethe, Tom Borchert, Eno Thereska, Borja Balle, and Neil Lawrence. 2018. Continual Learning in Practice. In *Proceedings of the Workshop on Continual Learning at NeurIPS*. https://doi.org/10.48550/ARXIV.1903.05202

[20] Alex Egg. 2021. Online Learning for Recommendations at Grubhub. In *Proceedings of the Conference on Recommender Systems (RecSys)*. https://doi.org/10.1145/3460231.3474599

[21] European Union. 2016. Art. 17 GDPR: Right to erasure ('right to be forgotten'). https://gdpr.eu/article-17-right-to-be-forgotten/.

[22] Clement Farabet and Nicolas Koumchatzky. 2020. Presentation: Inside NVIDIA's AI Infrastructure for Self-driving Cars. In *Presentations of the USENIX Conference on Operational Machine Learning (OpML)*. https://www.usenix.org/conference/opml20/presentation/farabet

[23] Feast Authors. 2023. Feast: Feature Store For Machine Learning. https://feast.dev/.

[24] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *Comput. Surveys* 46, 4 (2014), 1–37. https://doi.org/10.1145/2523813

[25] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, James Law, Kevin Lee, Jason Lu, Pieter Noordhuis, Misha Smelyanskiy, Liang Xiong, and Xiaodong Wang. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. https://doi.org/10.1109/HPCA.2018.00059

[26] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the International Workshop on Data Mining for Online Advertising (ADKDD)*. https://doi.org/10.1145/2648584.2648589

[27] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. 2020. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society* 146, 730 (2020). https://doi.org/10.1002/qj.3803

[28] Chip Huyen. 2020. Machine learning is going real-time. https://huyenchip.com/2020/12/27/real-time-machine-learning.html.

[29] Chip Huyen. 2022. *Designing Machine Learning Systems*. O'Reilly Media, Inc.

[30] Chip Huyen. 2022. Real-time machine learning: challenges and solutions. https://huyenchip.com/2022/01/02/real-time-machine-learning-challenges-and-solutions.html.

[31] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. 2021. Scalability vs. Utility: Do We Have to Sacrifice One for the Other in Data Importance Quantification?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr46437.2021.00814

[32] Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. 2020. Model Assertions for Monitoring and Improving ML Models. In *Proceedings of Machine Learning and Systems (MLSys)*. https://proceedings.mlsys.org/book/319.pdf

[33] Angelos Katharopoulos and François Fleuret. 2018. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. In *Proceedings of the International Conference on Machine Learning (ICML)*. http://proceedings.mlr.press/v80/katharopoulos18a.html

[34] Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. 2022. Online Continual Learning on Class Incremental Blurry Task Configuration with Anytime Inference. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=nrGGfMbY_qK

[35] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2021. WILDS: A Benchmark of in-the-Wild Distribution Shifts. In *International Conference on Machine Learning (ICML)*. https://proceedings.mlr.press/v139/koh21a.html

[36] Akinwande Komolafe. 2023. Retraining Model During Deployment: Continuous Training and Continuous Testing. https://neptune.ai/blog/retraining-model-during-deployment-continuous-training-continuous-testing.

[37] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. University of Toronto, Toronto, Ontario. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[38] Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomás Kociský, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. 2021. Mind the Gap: Assessing Temporal Generalization in Neural Language Models. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2021/hash/f5bf0ba0a17ef18f9607774722f5698c-Abstract.html

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998). https://doi.org/10.1109/5.726791

[40] Aodong Li, Alex Boyd, Padhraic Smyth, and Stephan Mandt. 2021. Detecting and Adapting to Irregular Distribution Shifts in Bayesian Online Learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2021/hash/362387494f6be6613daea643a7706a42-Abstract.html

[41] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M. van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German Ignacio Parisi, Fabio Cuzzolin, Andreas S. Tolias, Simone Scardapane, Luca Antiga, Subutai Ahmad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. 2021. Avalanche: An End-to-End Library for Continual Learning. In *Workshop Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*. https://doi.org/10.1109/CVPRW53098.2021.00399

[42] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient Episodic Memory for Continual Learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2017/hash/f87522788a2be2d171666752f97ddebb-Abstract.html

[43] Andrey Malinin, Neil Band, Yarin Gal, Mark J. F. Gales, Alexander Ganshin, German Chesnokov, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, Vatsal Raina, Vyas Raina, Denis Roginskiy, Mariya Shmatova, Panagiotis Tigas, and Boris Yangel. 2021. Shifts: A Dataset of Real Distributional Shift Across Multiple Large-Scale Tasks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS) (Benchmark Track)*, Joaquin Vanschoren and Sai-Kit Yeung (Eds.). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/ad61ab143223efbc24c7d2583be69251-Abstract-round2.html

[44] Ankur Mallick, Kevin Hsieh, Behnaz Arzani, and Gauri Joshi. 2022. Matchmaker: Data Drift Mitigation in Machine Learning for Large-Scale Systems. In *Proceedings of Machine Learning and Systems (MLSys)*. https://proceedings.mlsys.org/paper/2022/hash/1c383cd30b7c298ab50293adfecb7b18-Abstract.html

[45] Sören Mindermann, Jan Markus Brauner, Muhammed Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. 2022. Prioritized Training on Points that are Learnable, Worth Learning, and not yet Learnt. In *Proceedings of the International Conference on Machine Learning (ICML)*. https://proceedings.mlr.press/v162/mindermann22a.html

[46] Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. 2020. Coresets for Data-efficient Training of Machine Learning Models. In *Proceedings of the International Conference on Machine Learning (ICML)*. http://proceedings.mlr.press/v119/mirzasoleiman20a.html

[47] Akshay Naresh Modi, Chiu Yuen Koo, Chuan Yu Foo, Clemens Mewald, Denis M. Baylor, Eric Breck, Heng-Tze Cheng, Jarek Wilkiewicz, Levent Koc, Lukasz Lew, Martin A. Zinkevich, Martin Wicke, Mustafa Ispir, Neoklis Polyzotis, Noah Fiedel, Salem Elie Haykal, Steven Whang, Sudip Roy, Sukriti Ramesh, Vihan Jain, Xin Zhang, and Zakaria Haque. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. https://doi.org/10.1145/3097983.3098021

[48] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. (2019). https://doi.org/10.48550/ARXIV.1906.00091

[49] Neptune. 2023. Neptune.ai ML Metadata Store. https://neptune.ai/.

[50] NVIDIA. 2023. NVIDIA DLRM Example Implementation. https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Recommendation/DLRM. Accessed: 2023-03-08.

[51] NVIDIA. 2023. NVIDIA Triton Inference Server. https://developer.nvidia.com/nvidia-triton-inference-server. Accessed: 2023-03-09.

[52] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. 2022. Challenges in Deploying Machine Learning: A Survey of Case Studies. *Comput. Surveys* 55, 6 (2022). https://doi.org/10.1145/3533378

[53] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. 2022. FourCastNet:

A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators. https://doi.org/10.48550/ARXIV.2202.11214

[54] Robi Polikar, Lalita Upda, Satish S. Upda, and Vasant Honavar. 2001. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man and Cybernetics, Part C* 31, 4 (2001). https://doi.org/10.1109/5326.983933

[55] Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. 2022. Adaptive Second Order Coresets for Data-efficient Machine Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*. https://proceedings.mlr.press/v162/pooladzandi22a.html

[56] Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. 2020. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*. https://doi.org/10.1007/978-3-030-58536-5_31

[57] PyTorch Serve Contributors. 2020. TorchServe: Docs. https://pytorch.org/serve/. Accessed: 2023-03-09.

[58] Stephan Rabanser, Stephan Günnemann, and Zachary C. Lipton. 2019. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper/2019/hash/846c260d715e5b854ffad5f70a516c88-Abstract.html

[59] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.91

[60] Sebastian Schelter, Felix Bießmann, Tim Januschowski, David Salinas, Stephan Seufert, and Gyuri Szarvas. 2018. On Challenges in Machine Learning Model Management. *IEEE Data Engineering Bulletin* 41, 4 (2018). http://sites.computer.org/debull/A18dec/p5.pdf

[61] Shreya Shankar, Bernease Herman, and Aditya G. Parameswaran. 2022. Rethinking Streaming Machine Learning Evaluation. In *Proceedings of the ML Evaluation Standards Workshop at ICLR*. https://doi.org/10.48550/arXiv.2205.11473

[62] Chijun Sima, Yao Fu, Man-Kit Sit, Liyi Guo, Xuri Gong, Feng Lin, Junyu Wu, Yongsheng Li, Haidong Rong, Pierre-Louis Aublin, and Luo Mai. 2022. Ekko: A Large-Scale Deep Learning Recommender System with Low-Latency Model Update. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. https://www.usenix.org/conference/osdi22/presentation/sima

[63] State of California, USA. 2018. Section 1798.130 CCPA. https://ccpa-info.com/california-consumer-privacy-act-full-text/.

[64] Nils Strassenburg, Dominic Kupfer, Julia Kowal, and Tilmann Rabl. 2023. Efficient Multi-Model Management. In *Proceedings International Conference on Extending Database Technology, (EDBT)*. https://doi.org/10.48786/edbt.2023.37

[65] Nils Strassenburg, Ilin Tolovski, and Tilmann Rabl. 2022. Efficiently Managing Deep Learning Models in a Distributed Environment. https://doi.org/10.48786/EDBT.2022.12

[66] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. 2020. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR42600.2020.00252

[67] Abhijit Suprem, Joy Arulraj, Calton Pu, and Joao Ferreira. 2020. ODIN: Automated drift detection and recovery in video analytics. *Proceedings of the VLDB Endowment* 13, 12 (2020). https://doi.org/10.14778/3407790.3407837

[68] Ashraf Tahmasbi, Ellango Jothimurugesan, Srikanta Tirthapura, and Phillip B. Gibbons. 2021. DriftSurf: Stable-State / Reactive-State Learning under Concept Drift. In *Proceedings of the International Conference on Machine Learning (ICML)*. http://proceedings.mlr.press/v139/tahmasbi21a.html

[69] Tesla. 2019. Tesla Autonomy Day. https://www.youtube.com/watch?v=Ucp0TTmvqOE&t=6678s.

[70] Josh Tobin. 2021. Toward continual learning systems. https://gantry.io/blog/toward-continual-learning-systems/.

[71] Daniel Vela, Andrew Sharp, Richard Zhang, Trang Nguyen, An Hoang, and Oleg S. Pianykh. 2022. Temporal quality degradation in AI models. *Scientific Reports* 12, 1 (2022). https://doi.org/10.1038/s41598-022-15245-z

[72] Larysa Visengeriyeva, Anja Kammer, Isabel Bär, Alexander Kniesz, and Michael Plöd. 2023. MLOps Infrastructure Stack. https://ml-ops.org/content/state-of-mlops.

[73] Steve Wang and Will Cukierski. 2014. The Avazu Click-Through Rate Prediction Dataset. https://kaggle.com/competitions/avazu-ctr-prediction

[74] Weights&Biases. 2023. W&B: Dataset Versioning. https://docs.wandb.ai/guides/data-and-model-versioning/dataset-versioning#25c79f05-174e-4d35-abda-e5c238b8d6d6.

[75] Yufeng Cai Kaixu Ren Pengjie Wang Huimin Yi Yue Song Jing Chen Hongbo Deng Jian Xu Lin Qu Bo Zheng Wenbo Su, Yuanxing Zhang. 2022. GBA: A Tuning-free Approach to Switch between Synchronous and Asynchronous Training for Recommendation Models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*. https://proceedings.neurips.cc/paper_files/paper/2022/hash/be0a8ecf8b2743a4117557c5eca0fb79-Abstract-Conference.html

[76] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems* 135 (2022). https://doi.org/10.1016/j.future.2022.05.014

[77] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large Scale Incremental Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2019.00046

[78] Huaxiu Yao, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. 2022. Wild-Time: A Benchmark of in-the-Wild Distribution Shift over Time. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS) (Benchmark Track)*. https://openreview.net/forum?id=F9ENmZABB0

[79] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: a unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65. https://doi.org/10.1145/2934664

[80] Giovanni Zappella, Martin Wistuba, Lukas Balles, and Cedric Archambeau. 2022. Automatically retrain neural networks with Renate. https://aws.amazon.com/de/blogs/machine-learning/automatically-retrain-neural-networks-with-renate/.

[81] Mark Zhao, Niket Agarwal, Aarti Basant, Buğra Gedik, Satadru Pan, Mustafa Ozdal, Rakesh Komuravelli, Jerry Pan, Tianshu Bao, Haowei Lu, Sundaram Narayanan, Jack Langman, Kevin Wilfong, Harsha Rastogi, Carole-Jean Wu, Christos Kozyrakis, and Parik Pol. 2022. Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training. In *Proceedings of the Annual International Symposium on Computer Architecture (ISCA)*. https://doi.org/10.1145/3470496.3533044